SISTEMAS INERCIAIS PARA APLICAÇÃO AEROESPACIAL - SIA

Manual de operação do sistema de controle de atitude da mesa de mancal a ar de um eixo

SIA-1430-FAB-0015-00 **28.08.13**

REGISTRO DE ATUALIZAÇÕES

Data da primeira edição*: 28 de agosto de 2013* *SIA-1430-FAB-0015-00*

REVISÃO	DATA	PÁGINAS MODIFICADAS	RESPONSÁVEL
00	28.08.13	PRIMEIRA VERSÃO----------	

LISTA DE DISTRIBUIÇÃO

Data da primeira edição*: 28 de agosto de 2013* *SIA-1430-FAB-0015-00*

REVISÃO	DATA	NOME	PCT
00	28.08.13	INPE	SIA-SISCAO

SUMÁRIO

1 - INTRODUÇÃO

Este documento descreve as etapas necessárias para a configuração, ativação, operação e manutenção dos equipamentos instalados na mesa de mancal a ar de um eixo, instalada no laboratório de simulação LABSIM da Divisão de Mecânica Espacial e Controle (DMC) do Instituto Nacional de Pesquisas Espaciais (INPE).

2 - DOCUMENTOS, DEFINIÇÕES E ABREVIAÇÕES

2.1 Documentos de Referência

DR1 - CARRARA, V.; PADILHA, O. S.; VAROTTO, S. E. C.; RICCI, M. C. *Um experimento de teste da bobina de rotação do SCD2*. S. J. Campos, INPE, maio 1992 (INPE-5404-RPQ/658).

DR2 - CARRARA, V.; MILANI, P. G. Controle de uma mesa de mancal a ar de um eixo equipada com giroscópio e roda de reação. *V SBEIN - Simpósio Brasileiro de Engenharia Inercial*. Rio de Janeiro, Nov. 2007.

DR3 - CARRARA, V. Simulador de atitude. (Manual). Disponível em http://www2.dem.inpe.br/val/projetos/att_pro/, 2008.

DR4 - CARRARA, V. Comparação experimental entre formas de controle de atitude com rodas de reação. *VI SBEIN - Simpósio Brasileiro de Engenharia Inercial*. Rio de Janeiro, Nov. 2010(a).

DR5 - CARRARA, V. Experimental comparison between reaction wheel attitude controller strategies. *Journal of Aerospace Engineering, Sciences and Applications*, Vol 2, n. 2, 2010(b). p. 1-9. ISSN 2236-577X. Available in: http://www.aeroespacial.org.br/jaesa/editions/repository/v02/n02/01-Carrara.pdf.

DR6 - CARRARA, V.; SIQUEIRA, R.; OLIVEIRA, D. Speed and current mode strategy comparison in satellite attitude control with reaction wheels. *Proceedings of COBEM 2011 – 21st Brazilian Congress of Mechanical Engineering*. Natal, RN, Brazil, Oct. 2011.

DR7 - CARRARA, V.; KUGA, H. K. Estimação dos parâmetros de atrito em rodas de reação para controle de atitude. *Anais do VII SBEIN - Simpósio Brasileiro de Engenharia Inercial*. São José dos Campos, Nov. 2012.

DR8 - CARRARA, V.; KUGA, H. K. Estimating friction parameters in reaction wheels for attitude control. *XV International Symposium on Dynamic Problems of Mechanics, DINAME 2013. Proceedings*. Búzios, Feb. 2013(a). ISSN 2316-9567.

DR9 - CARRARA, V.; KUGA, H. K. Estimating Friction Parameters in Reaction Wheels for Attitude Control. *Mathematical Problems in Engineering*, Vol. 2013, Article ID 249674, 8 pages, 2013(b). DOI:10.1155/2013/249674

DR10 - ENGELBRECHT, J. A. A. *User's Manual for the SunSpace reaction wheel and gyroscope substistem*. SunSpace, Matieland, South Africa, 2005. (SS01-106000).

2.2 Abreviações

CTS	Clear to send
FOG	Fiber optics gyro
LABSIM	Laboratório de Simulação
RW	Reaction Wheel
RWSSIS	Reaction Wheel Sun Space and Information Systems
SAABT	Single Axis Air Bearing Table

SCD2 Satélite de Coleta de Dados 2

3 - DESCRIÇÃO DO SISTEMA

A mesa de mancal a ar de um eixo (SAABT – Single Axis Air Bearing Table) é um dispositivo que consiste de um mancal rotativo vertical sustentado por um mancal de encosto. Ambos são lubrificados por uma estreita camada de ar comprimido que flui por orifícios dispostos na parte fixa do mancal, ilustrados esquematicamente na Figura **1**. Assegura-se com isso que os torques de atrito do mancal sejam pequenos, da ordem de um milésimo de Newton-metro. Praticamente o único torque de atrito sofrido pela mesa é do tipo viscoso, causado pelo movimento da mesa que gira em contato com o ar atmosférico. Este tipo de atrito é proporcional à velocidade de rotação da mesa e, portanto, é muito pequeno quando a mesa estiver parada ou em velocidade baixa.

Esta mesa foi projetada e construída no INPE na década de 1980 e foi utilizada para testar alguns sistemas utilizados nos primeiros satélites nacionais. Em particular testou-se a bobina de eixo e sua eletrônica de acionamento que foram utilizadas no sistema de ajuste da velocidade angular do satélite SCD2 (**DR1**).

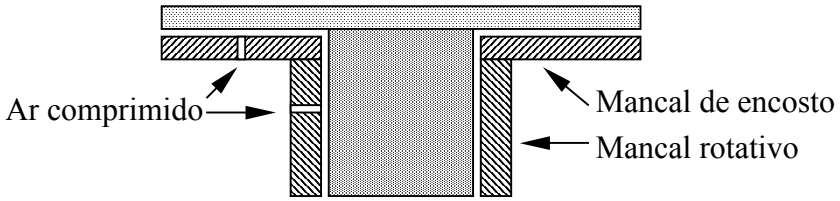

Fig. **1** – Esquema simplificado da mesa de mancal a ar.

Em 2007 a mesa foi equipada com uma roda de reação e um giroscópio de fibra ótica com eletrônica associada, todos produzidos pela empresa Sun Space, da África do Sul, denominados de RWSSIS (*Reaction Wheel Sun Space and Information Systems*). Com esta configuração implementou-se um controlador de atitude na mesa (**DR2**), além de diversos estudos do desempenho do controle e modelos matemáticos da roda de reação (**DR4**, **DR5**, **DR6**, **DR7**, **DR8**, **DR9**). A configuração atual da mesa é apresentada nas fotos das Figuras 2, 3 e 4. A mesa conta com, além da roda, giro e eletrônica, também com um conversor de interface serial bidirecional RS232-485, um rádio-modem que opera em interface serial RS232, uma bateria para alimentação, duas chaves mecânicas para ativação do sistema, um laser em cruz para verificação do apontamento, uma ventoinha de computador para gerar um pequeno torque de perturbação na mesa, um codificador ótico incremental (*encoder*) de 1200 pulsos por rotação, além de massas para balanceamento. Todos os componentes são apontados nas fotos das Figuras 2 a 4. Apresenta-se, na Figura 5, o esquema elétrico/eletrônico da mesa. As próximas seções irão descrever a operação e manutenção da mesa.

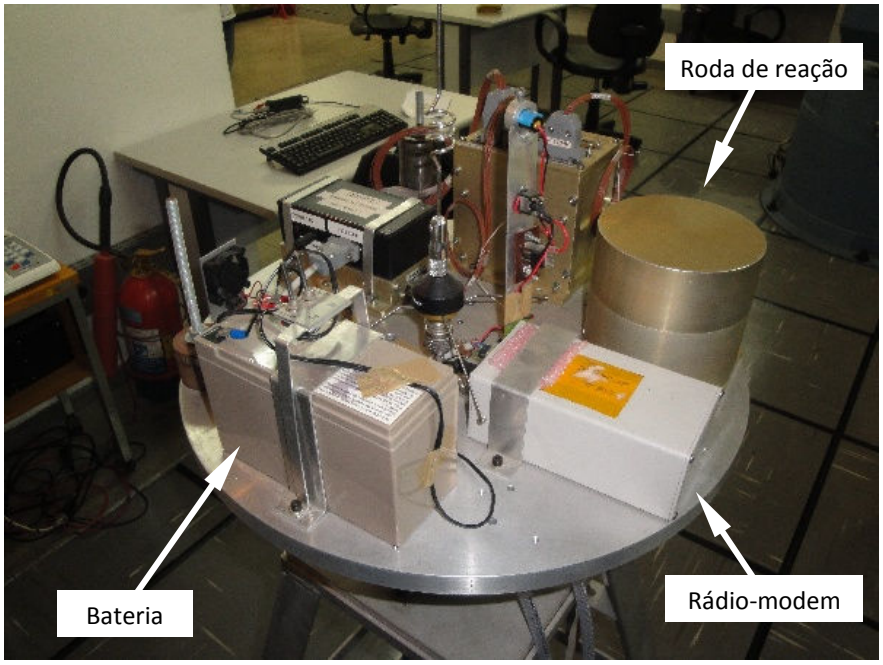

Fig. **2** – Mesa de mancal a ar de um eixo (roda de reação).

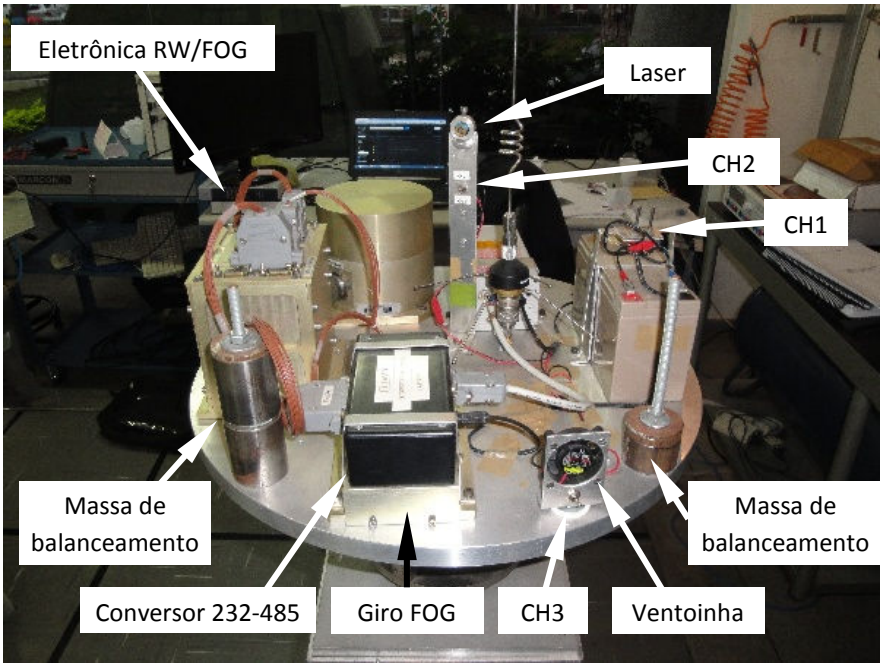

Fig. **3** – Mesa de mancal a ar de um eixo (giro FOG).

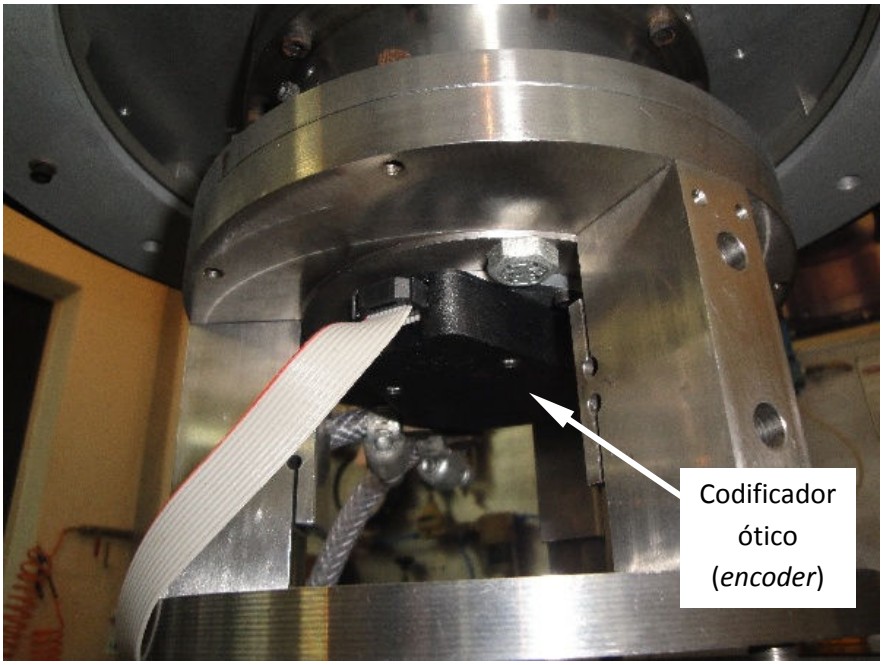

Fig. **4** – Mesa de mancal a ar de um eixo (*encoder*).

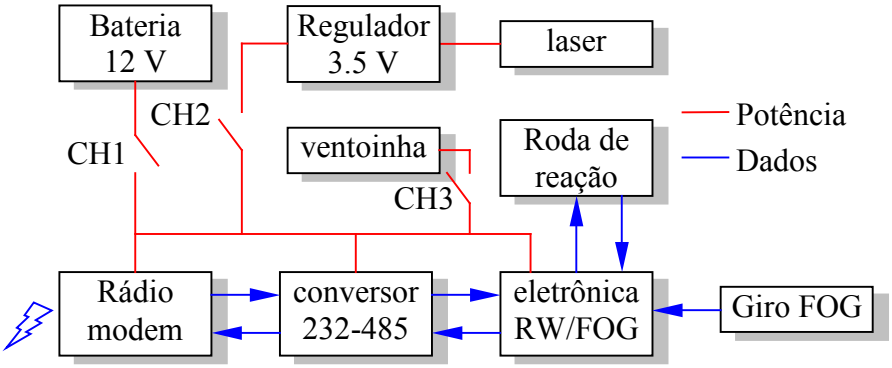

Fig. **5** – Esquema elétrico e eletrônico da mesa de mancal a ar de um eixo.

4 - OPERAÇÃO DO AR-COMPRIMIDO

A mesa de mancal a ar SAABT opera por princípio de flutuação sobre um colchão de ar-comprimido. A correta operação do ar-comprimido é essencial para garantir o funcionamento da mesa.

Quando o ar-comprimido está desligado, a mesa apóia-se num mancal de bronze localizado na parte estática, em contato com um disco de aço inoxidável (parte móvel), ambos polidos e planificados. É essencial evitar qualquer movimento da mesa quando o ar comprimido estiver desligado, para evitar que o aço risque o bronze.

O compressor de ar-comprimido encontra-se na parte externa do LABSIM e é ativado por uma chave térmica que fica no quadro de distribuição localizado no segundo piso do laboratório (Figura 6). Após o uso da mesa o compressor deve ser desligado.

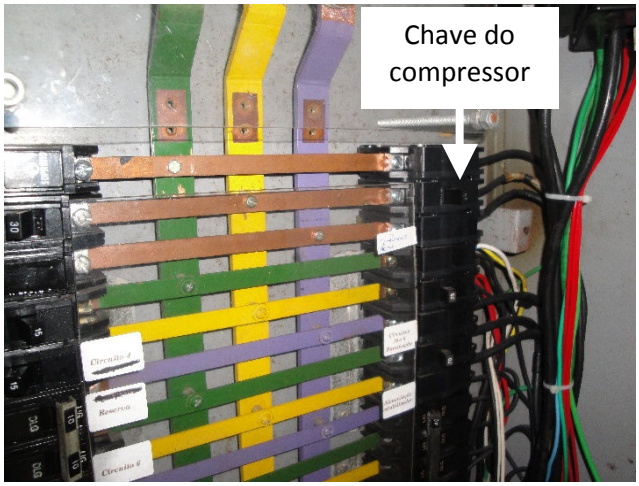

Fig. **6** – Quadro elétrico e chave de acionamento do ar-comprimido.

Após a compressão o ar passa por desumidificadores e filtros. Há um filtro localizado na parte externa que deverá ser descarregado de umidade sempre que for possível. Em seguida, a linha de ar-comprimido entra no laboratório e passa por uma válvula reguladora de pressão e um novo filtro desumidificador (regulador de entrada). Este conjunto válvula-filtro-regulador, mostrado na foto da Figura 7, deve ser regulado para liberar uma pressão de 8 a 9 bar. O filtro deve ser descarregado da umidade no início e no fim de operação da mesa. Em seguida o ar-comprimido passa pela bancada de ajuste de pressão, e irá alimentar as duas mesas de mancal a ar existentes no laboratório: a mesa de um eixo e a mesa de mancal semi-esférico. A bancada (Figura 8) conta com um reservatório na linha de entrada (não mostrado na foto, situado na parte inferior da bancada), um regulador de pressão principal, que atende as duas mesas, além de reguladores de pressão individuais: há dois deles para a mesa de mancal a ar de um eixo: um para o mancal rotativo e outro para o mancal de encosto. O regulador principal deve ser ajustado para uma pressão de cerca de 3 a 4 bar. Os dois reguladores da mesa de um eixo devem ser ajustados para uma pressão de 1,5 bar. Porém, o correto ajuste desta pressão obedece na verdade à massa embarcada no mancal. Recomenda-se que a pressão seja aumentada gradativamente nos dois reguladores, até que mesa perca contato físico, que se verifica imprimindo uma pequena força na mesa e observando se ela se desloca ou não sob ação da força. A partir deste ponto deve-se aumentar a pressão de 30 a 50% do seu valor como segurança para operação. O esquema pneumático da bancada, no que se refere à mesa de mancal a ar de um eixo, é mostrado na Figura 9, incluindo todos os reguladores.

A desativação do ar-comprimido, após os experimentos, segue caminho inverso. Inicialmente deve-se estancar o movimento rotacional da mesa, e, em seguida, elimina-se a pressão no mancal por meio dos dois reguladores de pressão. Elimina-se a pressão na linha por meio do regulador principal da bancada e desliga-se o compressor.

5 - CARGA DA BATERIA

A mesa de mancal a ar é equipada com uma bateria selada de 12 Volts de motocicleta. Esta bateria deve ser recarregada sempre que a tensão cair abaixo de 11 V. Quando a tensão estiver abaixo do necessário a roda de reação não obedece aos comandos enviados a ela,

mesmo que a eletrônica RW/FOG responda corretamente. Para carregar a bateria pode-se utilizar uma fonte de tensão e corrente controladas. A tensão deve ser ajustada a 12,7 V e a corrente deve ser limitada a 200 mA. Considera-se que a carga esteja completa quando a tensão atingir o valor nominal de 12,7 V. Pode-se igualmente utilizar uma eletrônica própria para carregar baterias, disponível no LABSIM. Neste caso a eletrônica indica por meio de um LED a carga completa da bateria. Recomenda-se que durante a carga evite-se operar a roda de reação e o giro, que devem permanecer desligados pela chave geral CH1.

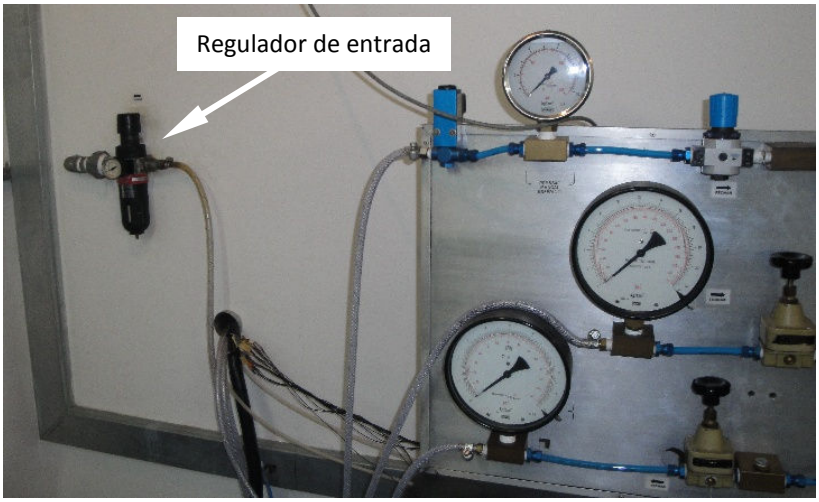

Fig. **7** – Regulador de pressão de entrada com filtro.

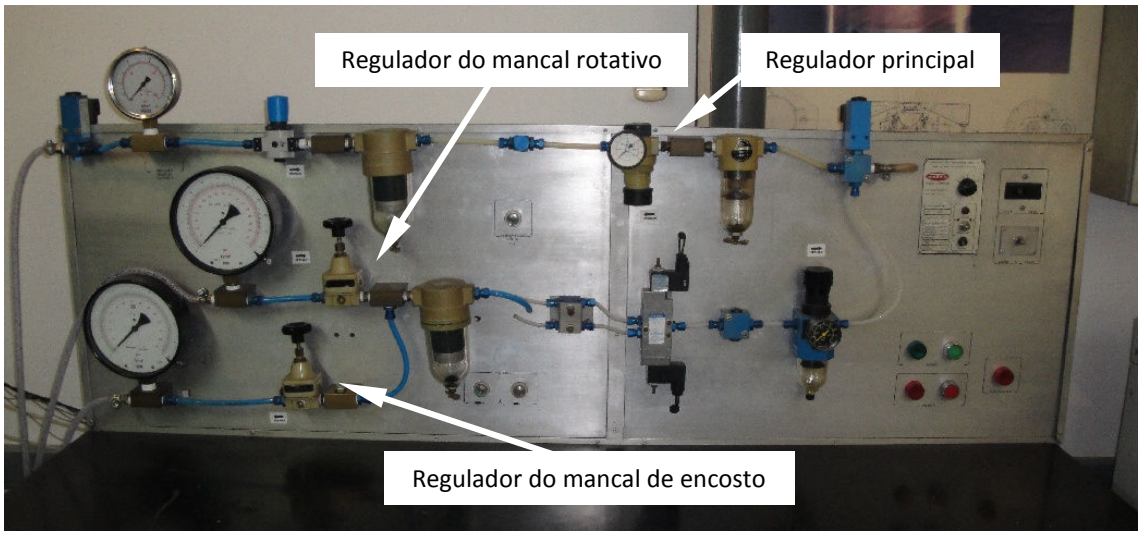

Fig. **8** – Bancada de ar-comprimido

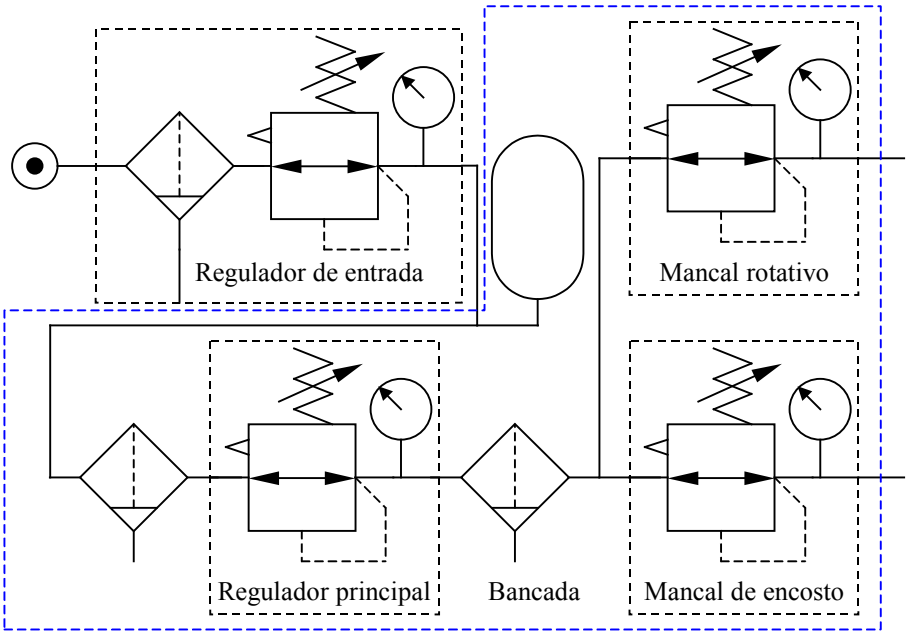

Fig. **9** – Esquema pneumático da bancada para alimentação das mesas de mancal a ar.

6 - BALANCEAMENTO

Devido ao fato do atrito ser bastante reduzido, a operação da mesa pode ser afetada por quaisquer desbalanceamentos e desnivelamentos que existam. De fato, por melhor que sejam o balanceamento e o nivelamento, não se pode garantir que sejam perfeitos, e, portanto, sempre haverá tanto um erro no nivelamento quanto um desbalanceamento residual. A minimização destes valores deve ser buscada por meio de técnicas especiais, relatadas aqui. O processo é feito em diversas etapas:

a) Inicia-se o processo de balanceamento e nivelamento provocando, propositadamente, um grande desnivelamento da mesa, por meio da mudança na posição de um dos parafusos de nivelamento. Um medidor de nível de precisão irá indicar este desnivelamento. Duas voltas completas em um dos parafusos de nivelamento, no sentido de abaixar a mesa neste local, são suficientes para promover o desnivelamento adequado, porém quanto maior o desnivelamento inicial melhor será o resultado do balanceamento a ser realizado a seguir.

b) Efetua-se a seguir, com a mesa desnivelada, um balanceamento por meio de acréscimo ou remoção de massa em um dos dois pontos existentes para isso, mostrados na Figura 3. A forma usual de realizar este balanceamento é observar o movimento da mesa flutuando sobre o colchão de ar-comprimido, e verificar se há um movimento pendular ao redor do parafuso usado para desnivelar. Procede-se ao ajuste de massas até que o movimento pendular não seja mais observado. Considera-se então que a mesa está balanceada. Este procedimento é bastante lento e pode durar várias horas.

c) Acrescenta-se, a seguir, uma massa de cerca de 1 kg, em um ponto qualquer da mesa, próximo da borda. Nesta situação a mesa se torna desbalanceada novamente e irá apresentar o movimento pendular, de tal forma que a massa irá tender para a posição do parafuso utilizado no desnivelamento.

d) Procura-se agora nivelar a mesa de forma a fazer com que o movimento pendular desapareça por completo. Para isso basta observar o ponto médio de oscilação pendular. Este ponto deverá ser erguido, ou, correspondentemente, o ponto oposto deve ser abaixado. Considera-se que a mesa esteja nivelada quando não for observado ou detectado o movimento pendular.

e) Remove-se a seguir a massa utilizada para o desbalanceamento, retornando a mesa na sua configuração de balanceada. O balanceamento e nivelamento estão agora completos.

7 - ATIVAÇÃO

A ativação do sistema de controle embarcado na mesa de mancal a ar é efetuado pela chave CH1, que liga simultaneamente a eletrônica da roda de reação e do giro FOG, além do conversor RS232-485 e do rádio-modem. O laser e a ventoinha são acionados por chaves próprias, mostradas na Figura 3. Após a ativação a eletrônica permanece em estado de espera, aguardando o envio de um telecomando para início de operação. Os telecomandos e as telemetrias disponíveis para o sistema são descritos na Seção 9.

8 - OPERAÇÃO

O conjunto RW/FOG pode ser operado de forma autônoma pela própria eletrônica de controle, sob requisição de telecomando, ou por meio de comandos diretos à roda ou de requisição de telemetria do giro e da roda. Os telecomandos e telemetrias serão descritos na próxima seção.

Quando operado de forma remota, usa-se o rádio-modem para fechar um laço de controle entre um computador externo e os instrumentos embarcados na mesa de mancal a ar. O rádio-modem possui dois transmissores-receptores gêmeos que operam em interface de comunicação serial RS232. Estes podem ser configurados para diferentes taxas de transmissão. Para maiores detalhes deve-se consultar o manual do fabricante.

Para realizar a comunicação usa-se um computador PC com interface RS232 com conexão de 9 pinos ou então um adaptador USB-RS232. Qualquer que seja a forma de comunicação, deve-se informar a porta de comunicação serial no programa que efetua a interface com a mesa. O número da porta de comunicação é estabelecido pelo gerenciador de dispositivos do Windows, em Portas COM. A função `start_comm` informa ao aplicativo o número da porta de comunicação serial associada ao modem. Veja-se os exemplos de uso da função no diretório Exemplos contido no arquivo com os *drivers* dos dispositivos. A figura 10 apresenta uma foto

do transmissor bidirecional radio-modem que deve ser conectado ao PC, seu cabo serial e sua fonte de alimentação.

Fig. **10** – Transmissor bidirecional radio-modem.

Na mesa de mancal a ar o rádio-modem comunica-se com a eletrônica do RWSSIS por meio de um conversor RS232-485, já que o sistema RWSSIS opera apenas com interface serial RS485, a uma taxa de 19200 bauds, 8 bits de dados, sem paridade, e 1 bit de parada. O conversor RS232-485 está configurado para operar na mesma velocidade, bem como o rádio-modem. O conversor foi fornecido pela Sun Space e não conta com manual. Por sua vez, o rádio-modem é de fabricação da empresa FreeWave, e, embora possa ser configurado para operar em velocidades maiores (sem que haja alteração na velocidade de comunicação na interface com o dispositivo escravo), não se recomenda efetuar tal alteração. O manual de operação do rádio-modem encontra-se disponível no diretório FreeWave que integra o arquivo que contém os diversos *drivers*. O manual traz detalhes que permitem configurar diversos parâmetros além da velocidade de comunicação.

Um codificador ótico (*encoder*) foi instalado na parte fixa da mesa de mancal a ar, de forma a se ter uma referência de posição. Não há contato físico entre o disco do codificador, que fica na parte girante, com a parte fixa. O *encoder* está fixado na parte inferior do mancal, e pode ser visto na Figura 4. A Figura 11 apresenta a eletrônica de interface entre o PC e o codificador ótico. Esta eletrônica opera em RS232, mas, da mesma forma que o radio-modem, pode-se também utilizar um conversor USB-RS232 na comunicação. A operação do codificador ótico, suas funções e configuração serão descritas na Seção 12.

Fig. **11** – Interface com o codificador ótico .

9 - FUNÇÕES PARA OPERAÇÃO DA RODA DE REAÇÃO E GIRO FOG

O fornecedor da roda de reação, do giro FOG e da eletrônica de controle de ambos, disponibilizou um *driver* para acionamento destes equipamentos, porém sem manual de uso. Felizmente as funções possuem um cabeçalho explicativo que, junto com o manual de comandos do sistema RWSSIS (**DR10**), permitem sua utilização sem dificuldades. O driver e os manuais estão disponibilizados para acesso no endereço:

http://www2.dem.inpe.br/val/projetos/rwexp/SAABT.zip.

Foram efetuadas algumas alterações no *driver* original, em virtude do fato de que os computadores atuais utilizam interface de comunicação USB ao contrário da interface RS232 necessária para operar o sistema. Em virtude disso, é necessário utilizar um adaptador USB para RS232, e este, normalmente, cria uma porta COM virtual com endereços compreendidos entre 2 e 14, sendo que o *driver* original admitia apenas portas entre 1 e 4. As funções que compõem o *driver* serão apresentadas a seguir, com uma pequena descrição da operação realizada. Mais informações podem ser obtidas no código fonte destas funções. Há cinco arquivos de códigos fonte:

- `RWSSISDemo.cpp` – Programa para demonstrar o uso do sistema
- `RWSSISDrivers.cpp` – Biblioteca de funções do *driver.*
- `RWSSISDrivers.h` – Cabeçalho de protótipos das funções do driver.
- `RWSSISComm.cpp` – Biblioteca de funções para tratamento de erros.
- `RWSSISComm.h` – Cabeçalho de protótipos de funções de comunicação serial.

O arquivo `RWSSISDemo.c` contém as seguintes funções

- **`void vSleep(unsigned int uiMilliseconds)`**

  Esta função é utilizada para deter o processamento um determinado intervalo de tempo.

- **`void main()`**

  Programa principal.

A biblioteca `RWSSISComm` contém funções que na versão original estavam distribuídas nos arquivos `RWSSISDemo.cpp` e `RWSSISDrivers.cpp`. Nesta versão esta biblioteca contém funções para tratamento de erros que possam acontecer na porta de comunicação. As funções que compõem esta biblioteca são:

- **`void vInitialiseErrorMessage(unsigned char ucResult);`**

  Esta função imprime o significado de códigos de erros de comunicação na porta serial.

- **`void vTelecommandErrorMessage(unsigned char ucResult);`**

  Esta função imprime os erros de transmissão de telecomando.

- **`void vTelemetryErrorMessage(unsigned char ucResult);`**

  Esta função imprime os erros que podem acontecer no recebimento de telemetria.

- **`void vFinaliseErrorMessage(unsigned char ucResult);`**

  Esta função imprime erros que porventura ocorram durante o fechamento da porta serial.

- **`int start_comm (int iter);`**

  Esta função estabelece e inicia a comunicação com o RWSSIS, a partir do número da porta serial armazenado na variável `iter` (por exemplo, se a comunicação for realizada pela porta COM4, então deve-se atribuir o valor 4 a `iter`). Esta função utiliza as funções `ucInitialiseComms`, `vInitialiseErrorMessage`, `ucSetRWGyroOnOff`, `vTelecommandErrorMessage`, e `ucFinaliseComms`.

- **`int end_comm (void);`**

  Esta função comanda a roda de reação para a velocidade nula, aguarda até que a roda de reação pare de girar enquanto imprime o valor da velocidade angular, comanda o desligamento tanto da roda quanto do giro FOG e encerra a comunicação serial.

O arquivo `RWSSISDrivers.cpp` contém funções para realizar leituras de telemetria e envio de telecomando para o RWSSIS. Descreve-se resumidamente a seguir as funções contidas nesta biblioteca. O significado dos parâmetros das funções pode ser visto no arquivo de cabeçalho `RWSSISComm.h`.

- **`unsigned char ucInitialiseComms(unsigned char ucComPort)`**

  Esta função abre uma porta de comunicação serial com o RWSSIS, definido pelo número da porta na variável `ucComPort`.

- **`unsigned char ucFinaliseComms(void)`**

  Esta função fecha a porta de comunicação serial.

- **unsigned char ucGetWheelSpeed(float *fSpeedMeasured)**

  Esta função efetua a leitura de telemetria da velocidade angular da roda de reação.

- **unsigned char ucGetArmatureCurrent(float *fCurrentMeasured)**

  Esta função efetua a leitura da corrente de armadura aplicada ao motor da roda de reação.

- **unsigned char ucGetGyroAccumulated(int *iGyroAccumulated)**

  Esta função faz a leitura do ângulo integrado pelo giro FOG desde a última leitura.

- **unsigned char ucGetGyroRaw(int *iGyroData, unsigned char *ucGyroStatus, unsigned char *ucGyroChecksum)**

  Esta função efetua a leitura da telemetria do giro FOG.

- **unsigned char ucGetStatus(unsigned char *ucControllerStatus, unsigned char *ucGyroStatus, unsigned char *ucWatchdogResetCounter, unsigned char *ucTripCounter)**

  Esta função verifica o estado dos componentes do RWSSIS.

- **unsigned char ucGetSoftwareVersion(unsigned char *ucMinorVersionNumber, unsigned char *ucMajorVersionNumber)**

  Esta função faz a leitura do indicador de versão do *firmware* armazenado na eletrônica de comando.

- **unsigned char ucGetRWTemperature(float *fRWTemperature)**

  Esta função recupera a temperatura do mancal da roda de reação.

- **unsigned char ucGetRWPressure(float *fRWPressure)**

  Esta função efetua a medida da pressão interna na roda de reação.

- **unsigned char ucGetInertialDeltaAngle(float *fAngleMeasured)**

  Esta função efetua a leitura do incremento de ângulo inercial (*inertial delta angle*) atual (veja-se a função `ucSetInertialDeltaAngle`).

- **unsigned char ucSetWheelSpeed(float fSpeedReference)**

  Esta função comanda a roda para uma dada velocidade angular em modo de controle de velocidade ou *speed mode control*.

- **unsigned char ucSetArmatureCurrent(float fCurrentReference)**

  Esta função comanda uma dada corrente de armadura no motor da roda de reação, em modo de controle por torque ou *torque mode control*.

- **unsigned char ucSetInertialRate(float fRateReference)**

Esta função faz com que a eletrônica do RWSSIS controle velocidade angular do satélite por meio de torques aplicados à roda de forma a manter nulo o erro entre a velocidade angular de referência `fRateReference` e aquela medida pelo giro.

- **`unsigned char ucSetInertialDeltaAngle(float fAngleReference)`**

  Esta função faz com que a eletrônica do RWSSIS controle a atitude por meio de torques aplicados à roda de forma a manter constante um determinado ângulo integrado pelo giro FOG.

- **`unsigned char ucSetRateGainK1(float fRateGainK1)`**

  Esta função modifica o valor do ganho de controle K_1 associado ao erro atual $e(k)$ quando a eletrônica controla a velocidade angular do satélite (veja-se a função `ucSetInertialRate`).

- **`unsigned char ucSetRateGainK2(float fRateGainK2)`**

  Esta função modifica o valor do ganho de controle K_2 associado ao erro anterior $e(k-1)$ quando a eletrônica controla a velocidade angular do satélite (veja-se a função `ucSetInertialRate`).

- **`unsigned char ucSetSpeedGainK1(float fSpeedGainK1)`**

  Esta função modifica o valor do ganho K_1 associado ao erro atual $e(k)$ no modo de controle de velocidade da roda de reação (veja-se a função `ucSetWheelSpeed`, e a Seção 10 adiante).

- **`unsigned char ucSetSpeedGainK2(float fSpeedGainK2)`**

  Esta função modifica o valor do ganho K_2 associado ao erro anterior $e(k-1)$ no modo de controle de velocidade da roda de reação (veja-se a função `ucSetWheelSpeed`, e a Seção 10 adiante).

- **`unsigned char ucSetAngleGainK1(float fAngleGainK1)`**

  Esta função modifica o valor do ganho K_1 associado ao erro de atitude no ângulo integrado pelo giro FOG. Veja-se a função `ucSetInertialDeltaAngle` e a Seção 8.4 do documento de interface do RWSSIS (**DR10**).

- **`unsigned char ucSetRWGyroOnOff(unsigned char ucRWOn, unsigned char ucGyroOn, unsigned char ucTripOverrideOn)`**

  Esta função permite ativar (ligar) ou desativar (desligar) tanto o giro FOG quanto a roda de reação de forma independente.

- **`unsigned char ucSetGyroBias(int iGyroBias)`**

  Esta função permite compensar o viés do giro FOG diretamente das medidas efetuadas.

Os arquivos de cabeçalho `RWSSISDrivers.h` e `RWSSISCom.h` contém apenas os protótipos das funções já descritas acima.

10 - CALIBRAÇÃO DO GIRO FOG

A operação da roda de reação não requer nenhuma configuração especial, porém o giro FOG exige que dois efeitos sejam corrigidos em suas medidas, para que o resultado possa ser utilizado numa malha de controle: o viés ou bias do giro, e a correção da velocidade de rotação da Terra.

A correção da velocidade de rotação da Terra é efetuado por meio da projeção da direção do eixo de rotação da Terra no eixo sensor do giro (eixo vertical local), o que leva à relação:

$$\omega_g = \omega_{ms} - \Omega_T \operatorname{sen}\varphi - b, \tag{1}$$

onde Ω_T é a velocidade angular da Terra, e vale

$$\Omega_T = \left(\frac{A_T + 1}{A_T}\right)\frac{360}{86400}\ ^\circ/\text{s} \approx 0{,}004178^\circ/\text{s}, \tag{2}$$

e no qual A_T é a duração do ano trópico em dias (A_T = 365,2422). φ é a latitude do local (φ = -23.21014444° para o LABSIM), b é o viés ou *bias*, e ω_{ms} é a velocidade angular transformada do giro. É importante observar que a telemetria apresenta as medidas do giro em unidades acumuladas de rotação, isto é, a leitura apresenta a rotação sofrida pelo giro desde a última medida, em unidades de giro, que corresponde a $8/32768^\circ \approx 0{,}00024414^\circ$, limitado a, no máximo, $\pm 8^\circ$. Desta forma, a velocidade angular transformada em unidades de graus por segundo pode ser obtida por

$$\omega_{ms} = \frac{8}{32768}\frac{\theta_{acc}}{\Delta t}\ ^\circ/\text{s}, \tag{3}$$

com Δt sendo o intervalo de tempo entre as medidas em segundos, e θ_{acc} é a medida acumulada do giro, fornecida pela função `ucGetGyroAccumulated`.

Para se corrigir o viés deve-se efetuar uma média de uma série de medidas com o giro parado e já corrigido da rotação da Terra. Assume-se que o viés esteja correto quando a média convergir para um valor fixo com pelo menos 4 dígitos significativos, o que demanda algumas horas a uma taxa de aquisição de 10 Hz, que é a taxa máxima suportada pelo sensor. O valor do viés deste giro foi estimado em $0{,}35\ 10^{-3}\ ^\circ/\text{s}$.

11 - OPERAÇÃO DA RODA DE REAÇÃO

A roda de reação do RWSSIS pode operar em modo de torque (ou de corrente), por meio da função `ucSetArmatureCurrent`, e em modo de velocidade, com a função `ucSetWheelSpeed`. Quando em modo de torque (corrente) atribui-se ao motor uma determinada corrente e com isso gera-se um torque, pois o torque em um motor é

aproximadamente proporcional à corrente. Já no modo de velocidade comanda-se a roda a manter uma dada velocidade angular. Nesta situação uma malha de controle interna ao RWSSIS calcula a corrente a ser aplicada à roda com base num controlador PI, dado por:

$$I_{ref}(t) = K_p\ e(t) + K_i \int_0^t e(t)\,dt\,, \tag{4}$$

no qual $e(k)$ é o erro na velocidade angular medida no instante t_k. Quando discretizado, este controle passa a ser expresso por

$$I_{ref}(k) = I_{ref}(k-1) + K_1\ e(k) - K_2\ e(k-1)\,, \tag{5}$$

e tal que $K_1 = p\,(K_p + K_i\ \Delta t)$, $K_2 = -p\ K_p$, Δt é o intervalo de tempo entre duas medidas sequenciais do erro e p é uma constante de proporcionalidade igual a 10723 rd/s/A, ou 102.4 rpm/mA, de acordo com o documento de interface do RWSSIS. Os ganhos podem ser alterados pelas funções `ucSetSpeedGainK1` e `ucSetSpeedGainK2`, mas seus valores *default* (quando o sistema é ativado) são K_1 = 2000 e K_2 = –1900. A telemetria da roda consiste na medida da sua velocidade angular, que pode ser amostrada a uma taxa máxima de 10 Hz, em unidades de rotações por minuto (rpm). A velocidade é fornecida pela função `ucGetWheelSpeed`. Pode-se também medir a corrente com a função `ucGetArmatureCurrent`, a temperatura do mancal com a função `ucGetRWTemperature`, e a pressão interna por meio da função `ucGetRWPressure`. A taxa máxima de amostragem de todas as telemetrias é de 10 Hz. Caso a telemetria seja inquirida a uma taxa maior do que 10 Hz haverá coincidência de valores amostrados.

Um exemplo de um programa que efetua o controle da mesa é apresentado na Seção 14. O código fonte deste programa é também disponibilizado junto com os demais códigos no link:

http://www2.dem.inpe.br/val/projetos/rwexp/SAABT.zip.

12 - FUNÇÕES DO CODIFICADOR ÓTICO

O codificador ótico (*encoder*) instalado na mesa de mancal a ar foi produzido pela empresa US Digital, do modelo E6, e fica localizado na parte inferior do mancal, como pode ser visto na Figura 4. A especificação do sensor é apresentada no seu *datasheet*, que pode ser encontrado na página do fabricante:

http://www.usdigital.com/products/encoders/incremental/ rotary/kit/E6.

A resolução do encoder é de 1200 pulsos por rotação, do tipo incremental, sem referência de ângulo inicial, isto é, o contador de pulsos perde a referência caso seja desligado. Contudo, o disco do codificador possui uma trilha de índice que contém um único pulso numa dada posição. Este índice, quando detectado, fornece uma posição fixa do codificador e pode ser utilizado para reposicionar o contador. O disco do codificador é fixado na parte rotativa do mancal e não mantém contato físico com qualquer parte fixa deste.

O sensor conta com uma eletrônica microprocessada que efetua a contagem de pulsos e provê a necessária comunicação com o computador por meio de interface serial RS232. O detector de pulsos fica na parte estática, o que significa que não é necessário utilizar um radio-modem ou qualquer meio sem-fio para efetuar a comunicação. Um cabo paralelo de 8 vias com cerca de 4 metros provê a comunicação entre o detector de pulsos e a eletrônica do encoder. Caso não se disponha de um computador com comunicação RS232, um adaptador USB-RS232 pode ser utilizado. O fabricante disponibiliza um *driver* em linguagem C para o sensor, mas, com a intenção de facilitar o uso deste *driver*, produziu-se uma nova versão com diversos aprimoramentos em relação ao original. Foram criados dois conjuntos de funções, idênticos na funcionalidade exceto pela inclusão de um parâmetro a mais em um dos conjuntos. Originalmente a comunicação pela interface serial requer a definição de uma variável do tipo `handle` (manipulador) que estabelece o canal e o *buffer* de comunicação. Em um dos conjuntos este manipulador é criado e armazenado internamente, e, portanto, não necessita ser especificado. Já o outro conjunto requer obrigatoriamente a criação do manipulador. Os nomes das funções em ambos os conjuntos permanecem iguais entre si, diferenciando-se pela inclusão do sufixo `_H` nas funções que requerem a criação prévia do manipulador. O motivo para a criação de dois conjuntos semelhantes de funções se deve ao uso distinto entre eles: no primeiro deles não é necessário qualquer especificação de um manipulador, o que facilita o uso das funções; o segundo caso tem aplicação quando dois ou mais codificadores óticos necessitam ser lidos simultaneamente no mesmo computador. Em resumo, utiliza-se o conjunto sem especificação do manipulador quando houver apenas um codificador ótico, ou o conjunto que necessita a declaração manipuladores quando houver mais do que um codificador ótico simultâneo.

Fazem parte do driver 3 arquivos, contidos no diretório `US Digital Driver`:

- `US_E6_Driver.cpp`, que contém o código de todas as funções,
- `US_E6_Driver.h`, que contém a definição de diversas variáveis de configuração e os protótipos das as funções, e
- `structures_PC.h`, que contém definições de estruturas utilizadas na comunicação com o sensor.

Infelizmente o fabricante do codificador não produziu um manual de uso do driver, o que dificulta seu uso e entendimento. Embora a maioria das funções seja de fácil compreensão, o pequeno cabeçalho explicativo no código fonte não é suficiente para seu entendimento completo. Em virtude disso apresenta-se aqui um manual de uso das funções, já quase totalmente testadas.

São descritas, a seguir, as principais funções do *driver* deste codificador, que permitem iniciar a leitura, configurar, fazer medições, etc. Para usá-lo é necessário a inclusão do cabeçalho `US_E6_Driver.h` no módulo principal. O *driver* e os demais arquivos de cabeçalhos podem ser obtidos no link:

http://www2.dem.inpe.br/val/projetos/rwexp/SAABT.zip.

As seguintes funções permitem abrir e configurar a porta de comunicação serial no computador que fará a interface com o codificador ótico:

- **`int US_E6_OpenCom(unsigned char ucComPort)`**

  A função `US_E6_OpenCom` abre uma porta de comunicação serial para uso do codificador ótico. A porta a ser aberta deve ser a mesma que foi atribuida pelo sistema operacional, que pode ser encontrada em Painel de Controle / Sistema / Gerenciador de Dispositivos / Portas COM (*Control Panel* / *System* / *Device Manager* / *COM Ports*). Caso um adaptador USB-RS seja utilizado, haverá uma porta serial virtual atribuída pelo sistema operacional. O manipulador é criado e armazenado internamente pela função **`US_E6_OpenCom`**.

Parâmetros de entrada:

`ucComPort`

Número da porta serial, do tipo `unsigned char`. Caso, por exemplo, a porta serial atribuída pelo sistema operacional seja COM4, então `ucComPort` deverá armazenar o valor 4. O valor da porta serial deve estar compreendido entre 1 e 14. Pode-se igualmente utilizar as variáveis `US_E6_COM1`, `US_E6_COM2`, ... `US_E6_COM14` definidas no arquivo de cabeçalho.

Parâmetros de saída:

`US_E6_OpenCom`

Indicador de erro do tipo `int`, com valores diferentes de zero caso ocorra algum problema na abertura da porta de comunicação, ou 0 se bem sucedido.

- **`HANDLE US_E6_OpenCom_H(unsigned char ucComPort)`**

  A função `US_E6_OpenCom_H` abre uma porta de comunicação serial para uso do codificador ótico. A porta a ser aberta deve ser a mesma que foi atribuida pelo sistema operacional, que pode ser encontrada em Painel de Controle / Sistema / Gerenciador de Dispositivos / Portas COM (*Control Panel* / *System* / *Device Manager* / *COM Ports*). Caso um adaptador USB-RS seja utilizado, haverá uma porta serial virtual atribuída pelo sistema operacional. O manipulador é criado e devolvido no retorno da função.

Parâmetros de entrada:

`ucComPort`

Número da porta serial, do tipo `unsigned char`. Caso, por exemplo, a porta serial atribuída pelo sistema operacional seja COM4, então `ucComPort` deverá armazenar o valor 4. O valor da porta serial deve estar compreendido entre 1 e 14. Pode-se igualmente utilizar as variáveis `US_E6_COM1`, `US_E6_COM2`, ... `US_E6_COM14` definidas no arquivo de cabeçalho.

Parâmetros de saída:

`US_E6_OpenCom`

Manipulador do tipo `HANDLE`.

- **int US_E6_CloseCom()**
- **int US_E6_CloseCom_H(HANDLE hCOMM)**

A função US_E6_CloseCom (ou US_E6_CloseCom_H) fecha a porta de comunicação serial que foi aberta pela função US_E6_OpenCom (ou US_E6_OpenCom_H). Esta função deve ser chamada ao término do programa que utiliza o codificador ótico.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

US_E6_CloseCom / US_E6_CloseCom_H

Indicador de erro do tipo int, com valores diferentes de zero caso ocorra algum problema no fechamento da porta de comunicação, ou 0 se bem sucedido.

- **int US_E6_ChangeLocalBaudRate(unsigned int baud)**
- **int US_E6_ChangeLocalBaudRate_H(HANDLE hCOMM, unsigned int baud)**

A função US_E6_ChangeLocalBaudRate (ou US_E6_ChangeLocalBaudRate_H) muda a velocidade de comunicação da interface serial do sistema operacional no computador, mas não altera a taxa de comunicação do codificador ótico. Para a correta operação do sensor, é necessário mudar antecipadamente a velocidade de comunicação do codificador, antes de alterá-la no sistema operacional. Para alterar a velocidade do codificador veja-se a função US_E6_ChangeBaud (ou US_E6_ChangeBaud_H).

Parâmetros de entrada:

baud

Velocidade de comunicação da interface serial. Pode assumir os valores: 1200, 2400, 4800, 9600, 19200, 38400, 57600 e 115200. Pode-se ainda utilizar os valores previamente definidos no cabeçalho US_E6_BR_1200, US_E6_BR_2400, US_E6_BR_4800, US_E6_BR_9600, US_E6_BR_19200, US_E6_BR_38400, US_E6_BR_57600 e US_E6_BR_115200.

Parâmetros de saída:

US_E6_ChangeLocalBaudRate / US_E6_ChangeLocalBaudRate_H

Indicador de erro do tipo int, com valores diferentes de zero caso ocorra algum problema na mudança da taxa de comunicação, ou 0 se bem sucedido.

- **int US_E6_SetTimeout(int tt)**
- **int US_E6_SetTimeout_H(HANDLE hCOMM, int tt)**

A função US_E6_SetTimeout (ou US_E6_SetTimeout_H) muda o tempo de espera da porta de comunicação serial. Ao se realizar uma leitura na porta serial, se o tempo

decorrido para o recebimento da informação ultrapassar o valor de `tt`, em mili-segundos, o processo de leitura é abandonado e retorna com um indicativo de erro. Esta função altera os parâmetros `ReadIntervalTimeout`, `ReadTotalTimeoutConstant`, e `WriteTotalTimeoutConstant` do manipulador da interface serial (veja-se a estrutura `COMMITIMEOUTS` da interface serial para mais informações). Em geral o valor do tempo de espera não necessita ser alterado, pois seu valor *default* é apropriado.

Parâmetros de entrada:

`tt`

Intervalo de tempo em mili-segundos do tempo de espera.

Parâmetros de saída:

`US_E6_SetTimeout` / `US_E6_SetTimeout_H`

Indicador de erro do tipo `int`, com valores diferentes de zero caso ocorra algum problema na mudança do tempo de espera, ou 0 se bem sucedido.

- **`void US_E6_Purge()`**
- **`void US_E6_Purge_H(HANDLE hCOMM)`**

A função `US_E6_Purge` (ou `US_E6_Purge_H`) elimina qualquer informação provinda da interface serial e que ainda se encontra acumulada no *buffer* da interface. Qualquer informação presente no *buffer* será descartada.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

Não há.

As funções descritas a seguir permitem iniciar e configurar a comunicação com o codificador ótico:

- **`int US_E6_ChangeBaud(unsigned int baud)`**
- **`int US_E6_ChangeBaud_H(HANDLE hCOMM, unsigned int baud)`**

A função `US_E6_ChangeBaud` (ou `US_E6_ChangeBaud_H`) altera a taxa de comunicação com o codificador ótico e também muda a velocidade de comunicação da interface serial do sistema operacional no computador. Para alterar apenas a velocidade de comunicação no sistema operacional veja-se a função `US_E6_ChangeLocalBaudRate` (ou `US_E6_ChangeLocalBaudRate_H`).

Parâmetros de entrada:

`baud`

Velocidade de comunicação da interface serial. Pode assumir os valores: 1200, 2400, 4800, 9600, 19200, 38400, 57600 e 115200. Pode-se ainda utilizar os valores previamente definidos no cabeçalho `US_E6_BR_1200`, `US_E6_BR_2400`, `US_E6_BR_4800`, `US_E6_BR_9600`, `US_E6_BR_19200`, `US_E6_BR_38400`, `US_E6_BR_57600` e `US_E6_BR_115200`.

Parâmetros de saída:

`US_E6_ChangeBaud` / `US_E6_ChangeBaud_H`

Indicador de erro do tipo `int`, com valores diferentes de zero caso ocorra algum problema na mudança da taxa de comunicação, ou 0 se bem sucedido.

- **`int US_E6_Ping()`**
- **`int US_E6_Ping_H(HANDLE hCOMM)`**

A função `US_E6_Ping` (ou `US_E6_Ping_H`) verifica a presença e a correta configuração do codificador ótico na interface serial. A função retorna com o valor 1 se foi detectado o codificador ótico ou 0 caso contrário.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`US_E6_Ping` / `US_E6_Ping_H`

Indicador de retorno: 1 se foi detectado o codificador ótico ou 0 caso contrário.

- **`int US_E6_Initialize(int quadmode, int preset)`**
- **`int US_E6_Initialize_H(HANDLE hCOMM, int quadmode, int preset)`**

A função `US_E6_Initialize` (ou `US_E6_Initialize_H`) inicia o codificador ótico para operação de leitura e configura a quadratura e o valor de comparação. Veja-se também as funções `US_E6_SetMultiplier`, `US_E6_SetCompare` e `US_E6_PassedCompare` (`US_E6_SetMultiplier_H`, `US_E6_SetCompare_H` e `US_E6_PassedCompare_H`).

Parâmetros de entrada:

`quadmode`

Quadratura a ser selecionada no codificador ótico. Pode assumir os valores: 0, 1, 2 e 4. Pode-se igualmente utilizar os valores `US_E6_QUADMODE_0`, `US_E6_QUADMODE_1`, `US_E6_QUADMODE_2`, `US_E6_QUADMODE_4`, todos definidos no arquivo de cabeçalho. Quadratura 0 faz com que o codificador fique inativo, isto é, não apresente medidas. Quadratura igual a 1 habilita o codificador a medir a cada rampa de subida do canal A, resultando em 1200 pulsos a cada rotação do eixo no presente modelo US_R6. Uma quadratura de

2 faz com que o codificador incremente o contador tanto na subida quanto na descida do canal A, portanto com resolução de 2400 pulsos por ciclo. A quadratura de 4 permite medidas tanto no canal A quanto no canal B, em ambos os sentidos, resultando numa resolução de 4800 pulsos por rotação. Qualquer outro valor de `quadmode` será considerado como sendo de quadratura igual a 1.

`preset`

Valor a ser armazenado no codificador ótico e que permite que este gere um sinal na linha CTS (*Clear To Send*) da interface serial quando seu contador for igual ao valor de `preset`.

Parâmetros de saída:

`US_E6_Initialize` / `US_E6_Initialize_H`

Indicador de retorno: 0 ser bem sucedido ou 1 caso o valor de `quadmode` seja inválido.

- **`int US_E6_SetMultiplier(int quadmode)`**
- **`int US_E6_SetMultiplier_H(HANDLE hCOMM, int quadmode)`**

A função `US_E6_SetMultiplier` (ou `US_E6_SetMultiplier_H`) configura a quadratura do codificador ótico.

Parâmetros de entrada:

`quadmode`

Quadratura a ser selecionada no codificador ótico. Pode assumir os valores: 0, 1, 2 e 4. Veja-se o parâmetro `quadmode` na função `US_E6_Initialize` para mais detalhes.

Parâmetros de saída:

`US_E6_SetMultiplier` / `US_E6_SetMultiplier_H`

Indicador de retorno: 0 ser bem sucedido ou 1 caso o valor de `quadmode` seja inválido.

- **`void US_E6_SetCompare(int compareval)`**
- **`void US_E6_SetCompare_H(HANDLE hCOMM, int compareval)`**

A função `US_E6_SetCompare` (ou `US_E6_SetCompare_H`) atribui o valor de comparação de posição ao codificador ótico. Quando o contador passar pelo valor armazenado em `compareval` um sinal na linha CTS será acionado. Veja-se também o parâmetro `preset` da função `US_E6_Initialize` (ou `US_E6_Initialize_H`) e a função `US_E6_PassedCompare` (ou `US_E6_PassedCompare_H`).

Parâmetros de entrada:

`compareval`

Valor de comparação de posição angular do codificador ótico. O codificador ótico ativa a linha CTS quando o seu contador interno passar pelo valor atribuído por esta função. A informação de passagem ou não será recuperada pela função `US_E6_PassedCompare` (ou `US_E6_PassedCompare_H`).

Parâmetros de saída:

Não há.

- **`int US_E6_PassedCompare(int comport)`**
- **`int US_E6_PassedCompare_H(HANDLE hCOMM, int comport)`**

A função `US_E6_PassedCompare` (ou `US_E6_PassedCompare_H`) retorna com uma condição do codificador ótico haver cruzado ou não o valor de referência atribuído pelas funções `US_E6_SetCompare` (ou `US_E6_SetCompare_H`), ou `US_E6_Initialize` (ou `US_E6_Initialize_H`). Quando o contador do codificador passar pelo valor de referência, um sinal na linha CTS (*Clear To Send*) será acionado, que será então lido por esta função. Devido a características construtivas da função, é necessário passar como argumento o número da porta de comunicação, que é, de certa forma, redundante com relação ao manipulador. Obviamente o número da porta deve ser igual àquele associado ao manipulador criado pela função `US_E6_OpenCom` (ou `US_E6_OpenCom_H`).

Parâmetros de entrada:

`ucComPort`

Número da porta serial, do tipo `unsigned char`. O valor da porta serial deve estar compreendido entre 1 e 14. Pode-se igualmente utilizar as variáveis `US_E6_COM1`, `US_E6_COM2`, ... `US_E6_COM14` definidas no arquivo de cabeçalho.

Parâmetros de saída:

`US_E6_PassedCompare` / `US_E6_PassedCompare_H`

Indicador de retorno: 1 se ocorreu a passagem, 0 caso contrário, e o valor atribuído pela função `GetLastError` (do próprio sistema operacional) caso ocorra algum erro.

- **`void US_E6_ResetCounter()`**
- **`void US_E6_ResetCounter_H(HANDLE hCOMM)`**

A função `US_E6_ResetCounter` (ou `US_E6_ResetCounter_H`) reinicia o contador do codificador ótico a partir da posição atual. O contador interno do codificador possui um contador de 4 bytes (32 bits), capaz de gerar cerca de 2 bilhões de pulsos positivos e mais 2 bilhões negativos, antes que exceda sua capacidade. Isto é equivalente a mais de 880 mil revoluções completas com máxima quadratura. Contudo, mesmo sendo um valor muito grande ele pode ser eventualmente ultrapassado, e, neste, caso, o contador inverte seu

sinal, passando de 2147483647 para −2147483648 (ou vice-versa) num único incremento, o que pode causar problemas num sistema de controle. Esta função permite, então, que o contador possa ser re-iniciado sob supervisão do utilizador sempre que necessário, para evitar a dependência de um evento externo. Uma outra forma de evitar o estouro do contador é permitindo a habilitação do indicador (veja-se a função `US_E6_EnableIndex`, ou `US_E6_EnableIndex_H`)

Parâmetros de entrada:

Não há.

Parâmetros de saída:

Não há.

- **`int US_E6_EnableIndex()`**
- **`int US_E6_EnableIndex_H(HANDLE hCOMM)`**

A função `US_E6_EnableIndex` (ou `US_E6_EnableIndex_H`) habilita o reinício do contador do codificador ótico sempre que for encontrado o indicador na trilha do índice. Este indicador é uma marca gravada no disco do codificador, e que gera um pulso em um terceiro canal (além dos canais A e B) uma vez a cada revolução, sempre na mesma posição angular. O valor do contador, neste caso, fica confinado ao intervalo −4800 a 4800, no caso da máxima quadratura. Quando o contador ultrapassa o valor máximo ou cai abaixo do valor mínimo ele é reiniciado com valor nulo. Veja-se também a função `US_E6_DisableIndex` (ou `US_E6_DisableIndex_H`).

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`US_E6_EnableIndex` / `US_E6_EnableIndex_H`

Indicador de retorno: 0 se foi bem sucedido ou 1 caso tenha ocorrido algum erro.

- **`int US_E6_DisableIndex()`**
- **`int US_E6_DisableIndex_H(HANDLE hCOMM)`**

A função `US_E6_DisableIndex` (ou `US_E6_DisableIndex_H`) desabilita o reinício do contador do codificador ótico. Veja-se a função `US_E6_EnableIndex` (ou `US_E6_EnableIndex_H`) para mais detalhes.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`US_E6_DisableIndex` / `US_E6_DisableIndex_H`

Indicador de retorno: 0 se foi bem sucedido ou 1 caso tenha ocorrido algum erro.

- **`void US_E6_SetPosition(int setval)`**
- **`void US_E6_SetPosition_H(HANDLE hCOMM, int setval)`**

A função `US_E6_SetPosition` (ou `US_E6_SetPosition_H`) permite ajustar o valor corrente do contador interno do codificador ótico. Deve ser mencionado que a alteração do contador afeta eventuais comparações realizadas por meio da função `US_E6_PassedCompare` (ou `US_E6_PassedCompare_H`), que necessita, por esta razão, ser reiniciada com novo valor de referência.

Parâmetros de entrada:

`setval`

Valor a ser atribuído ao contador, passado por valor, do tipo `int`.

Parâmetros de saída:

Não há.

Apresentam-se, a seguir, as descrições das funções que efetuam medidas angulares do codificador ótico.

- **`long US_E6_ReadPosition()`**
- **`long US_E6_ReadPosition_H(HANDLE hCOMM)`**

A função `US_E6_ReadPosition` (ou `US_E6_ReadPosition_H`) efetua uma medida no valor corrente do contador do codificador ótico, em relação à posição inicial codificador (quando foi ligado), ou em relação à posição na qual ocorreu a última chamada da função `US_E6_ResetCounter` (ou `US_E6_ResetCounter_H`). Se o indicador de índice estiver ativo (veja-se a função `US_E6_EnableIndex`, ou `US_E6_EnableIndex_H`), então o contador será anulado quando o codificador cruzar o indicador, o que ocorre num valor indeterminado do contador durante a primeira revolução. O número de contagens por revolução depende da configuração da quadratura (veja-se as funções `US_E6_Initialize` ou `US_E6_Initialize_H`, e `US_E6_SetMultiplier` ou `US_E6_SetMultiplier_H`).

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`US_E6_ReadPosition` / `US_E6_ReadPosition_H`

Valor corrente do contador, do tipo `long`. Este tipo de inteiro possui também 4 bytes e é semelhante ao `int` nas implementações do MS Visual Studio. A real

posição angular do codificador depende da posição inicial ou da habilitação do indicador de índice.

- **`double US_E6_ReadAngleDeg()`**
- **`double US_E6_ReadAngleDeg_H(HANDLE hCOMM)`**

A função `US_E6_ReadAngleDeg` (ou `US_E6_ReadAngleDeg_H`) foi criada para permitir a leitura angular da posição do codificador ótico. O ângulo é medido em relação à posição inicial codificador (quando foi ligado), ou em relação à posição na qual ocorreu a última chamada da função `US_E6_ResetCounter` (ou `US_E6_ResetCounter_H`). Se o indicador de índice estiver ativo (veja-se a função `US_E6_EnableIndex`, ou `US_E6_EnableIndex_H`), então o ângulo será anulado quando o codificador cruzar o indicador, o que ocorre para um ângulo indeterminado durante a primeira revolução. Esta função ajusta automaticamente o ângulo com relação à quadratura empregada, porém o ângulo será errôneo se a quadratura for alterada após o início de operação. Neste caso recomenda-se alterar a quadratura apenas quando o codificador estiver na posição angular nula.

Parâmetros de entrada:

Não há.

Parâmetros de saída:

`US_E6_ReadPosition` / `US_E6_ReadPosition_H`

Valor corrente do ângulo, do tipo `double`. A real posição angular do codificador depende da posição inicial ou da habilitação do indicador de índice.

As rotinas do *driver* do codificador ótico foram testadas por meio de um adaptador FTDI USB-RS323 em Windows e com compilador Microsoft Visual C++ 2008 Express Edition, e todas tiveram comportamento correto, exceto as funções `US_E6_SetCompare`, `US_E6_PassedCompare` e suas equivalentes, que não apresentaram o resultado esperado em virtude da interface FTDI não possuir a linha de CTS necessária para a transmissão da informação.

Com a finalidade de averiguar visualmente o funcionamento do sistema de controle de atitude baseado no giro FOG e na roda de reação, foi instalado na mesa um apontador laser que projeta no ambiente um feixe em forma de cruz. Por meio de um painel (alvo) fixado na parede do laboratório, consegue-se avaliar o erro de apontamento do controlador, bem como as características da resposta a impulsos e desvios em regime permanente sob ação de torques provocados pela ventoinha. Porém, a origem deste alvo não se encontra na posição do índice contido no codificador ótico. Isto significa que, quando o codificador se encontra na posição nula o laser está fora do alvo e vice-versa. Uma vez que é possível desabilitar a detecção do índice e definir a posição da origem no codificador ótico por meio da função `US_E6_SetPosition`, então se pode criar um método para fazer com que as origens sejam coincidentes. Um possível algoritmo é mostrado na Figura 12. O algoritmo verifica a cada décimo de segundo se houve um cruzamento com o índice do codificador. Se o codificador

cruzar o índice, então um novo ciclo é iniciado e assim a medida do ângulo sofre uma mudança de 360 graus.

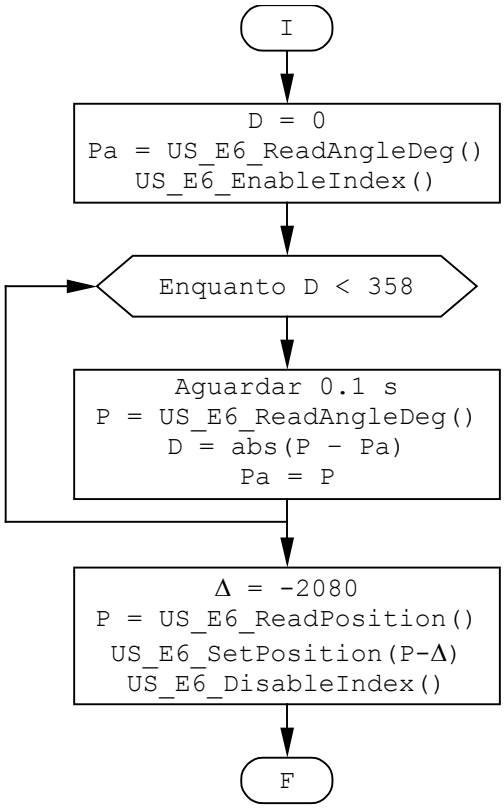

Fig. **12** – Fluxograma para correção da origem nas medidas do codificador ótico.

Ressalta-se que o codificador inicia-se sempre com medida angular nula na posição em que foi acionado, que é diferente, em geral, da posição em que se encontra o índice. Por isso é necessário fazer com que execute uma ou mais voltas colocando-se a mesa a girar em qualquer direção, para que a detecção do índice ocorra. Após a detecção do índice deve-se posicionar o contador para as origens coincidam. O desvio angular, no caso de quadratura 4 e para a atual posição do alvo do laser, corresponde a uma contagem de –2080 do contador. A seguir deve-se desabilitar o índice para que não ocorra mudança de quadrante quando o codificador cruzá-lo.

Com base neste algoritmo criou-se uma função, também inserida no material disponibilizado para uso do SAABT, que ajusta a posição inicial da origem do contador. Esta função, `US_E6_Home(long lDelta)` interage com o operador por meio de um terminal console e solicita que a mesa seja posta a girar, e, depois de detectado o índice, que a mesa seja bloqueada por alguns instantes para que o ajuste seja realizado. O argumento `lDelta` fornece o valor da origem do alvo em termos de posição angular do contador (`lDelta` = –2080). Mais detalhes a respeito da função podem ser encontrados no código fonte do programa `encoder.cpp`.

13 - FUNÇÕES DE TEMPO-REAL

Para operar os equipamentos instalados na mesa, como o giroscópio FOG, a roda de reação e o codificador ótico é necessário contar com alguma forma de sincronismo das atividades ao longo do tempo. Para isso pode-se utilizar pacotes já prontos para efetuar sincronismo e permitir a execução em tempo-real. Há duas formas de se deter o processamento enquanto que se aguarda a ocorrência de um evento no tempo: a hibernação ou a malha de retenção. Na hibernação configura-se um determinado instante de tempo no futuro para que o sistema operacional "desperte" o programa da hibernação. A partir daí o processo ou programa é posto para hibernar pelo sistema operacional sob requisição do próprio programa. Até que o programa entre em execução novamente, o sistema operacional poderá executar diversas outras atividades. Na malha de retenção, mais simples de ser executada, aciona-se uma malha cuja única forma de liberação é a ocorrência de um evento temporal. Em geral este evento é uma simples comparação entre o instante de sincronização e o tempo assinalado pelo relógio interno do computador.

Cabe ao programador decidir sobre a melhor forma de sincronizar as atividades, porém disponibilizou-se a biblioteca `rt` desenvolvida por Carrara (**DR3**). O manual de uso desta biblioteca pode ser encontrado no *link* fornecido na referência bibliográfica. O código fonte desta biblioteca é fornecido junto com os demais *drivers* dos equipamentos no *link*:

http://www2.dem.inpe.br/val/projetos/rwexp/SAABT.zip.

14 - EXEMPLO DE UM PROGRAMA DE CONTROLE EM C

A operação e controle da mesa de mancal a ar de um eixo (SAABT) pode ser realizada com o uso de qualquer linguagem computacional, como C++, Fortran e Matlab. Contudo, todos os exemplos vistos neste documento usam a linguagem C++, pois os *drivers* para comunicação são escritos nesta linguagem. Embora seja perfeitamente possível combinar estes *drivers* com as demais linguagens, a interface entre eles já não é imediata, devendo-se ter atenção com os parâmetros de entrada e de retorno das funções. Deixa-se a cargo do utilizador uma eventual adaptação dos recursos disponibilizados aqui para uma outra linguagem, como o Matlab.

Utilizou-se, no programa de exemplo do controlador de atitude, o compilador MS Visual C++ Express Ediction, facilmente encontrado no sítio da Microsoft, e de uso liberado gratuitamente. O programa é composto pelo programa principal `main` e a função `rwssis_example`, ambos presentes no código `Exemplo.cpp`, que efetuam o controle de atitude da mesa usando o modo de controle em velocidade angular da roda (veja-se a Seção 11). A atitude é obtida por meio da integração da velocidade angular da plataforma, medida pelo giroscópio FOG. Uma vez que não há um sensor de atitude (neste caso optou-se por não utilizar o codificador ótico), então a plataforma necessita estar direcionada na atitude de referência ao se iniciar o controle. O próprio programa informa este fato, por meio da impressão numa tela de console, que a mesa deve ser orientada na posição correta.

Quando operada no modo de corrente, o torque gerado pela roda de reação é aproximadamente proporcional à corrente, ou seja:

$$T_w = k_m \, i_w \,, \tag{6}$$

onde i_w é a corrente de armadura no motor. A constante de proporcionalidade k_m foi medida de forma indireta por meio de experimentos (**DR2**, **DR4**, **DR5**, **DR7**, **DR8**, **DR9**), resultando no valor 0,0228 Nm/A. Num outro experimento, a atitude foi controlada em corrente por meio de um controlador PID cujos ganhos foram ajustados por tentativas, e chegou-se aos valores $k_p = 0{,}04$ A/°, $k_d = 0{,}2$ A s/° (tempo derivativo $T_d = 5$ s) e $k_i = 0{,}001$ A/°s (tempo integral $T_i = 40$ s), e a corrente foi calculada por

$$i_w(k) = k_p \sum_{i=1}^{k} \omega_g(i) + k_i \sum_{i=1}^{k} \sum_{j=1}^{i} \omega_g(j) + k_d \, \omega_g(i) \,, \tag{7}$$

na qual $\omega_g(i)$ é a velocidade angular medida pelo giroscópio, já corrigida do viés e da rotação da Terra, no instante discretizado $t_i = i \, \Delta t$. Deseja-se que, ao se utilizar o modo de controle em velocidades, os torques comandados ou, analogamente, os ganhos do controlador, possam ser comparados com aqueles gerados no controlador operando em corrente. Porém, quando operado em velocidade o torque pode ser aproximado por:

$$T_w \cong J_w \, \dot{\omega}_w \,, \tag{8}$$

onde ω_w é a velocidade angular da roda, fornecida por telemetria, e J_w é o momento de inércia da roda, que vale 1.5 10^{-3} kg m^2, segundo o fabricante (**DR10**). A derivada desta velocidade angular pode ser calculada por derivada numérica, já que o ruído presente nesta medida é pequeno, por se tratar de um codificador ótico. Assim, substituindo-se a expressão do torque em função da corrente nesta última, tem-se que o incremento de velocidade angular passa a ser calculado por

$$\omega_w(k) = C \, i_w(k) + \omega_w(k-1) \,, \tag{9}$$

onde

$$C = \frac{k_m}{J_w} \Delta t \,, \tag{9}$$

com i_w, agora, fornecido pela expressão do controlador PID. Nota-se que, na expressão acima, $\omega_w(k-1)$ é a velocidade angular medida pela roda de reação, enquanto que $\omega_w(k)$ será a velocidade a ser comandada (velocidade de referência). Embora esta expressão ainda apresente ruídos em virtude das medidas da velocidade angular, tanto a roda quanto a plataforma se comportam como se fossem filtros passa-baixas, em virtude de suas grandes inércias quando comparadas com a frequência de atuação do controlador. O fluxograma simplificado da função `rwssis_example` é apresentado na Figura 13.

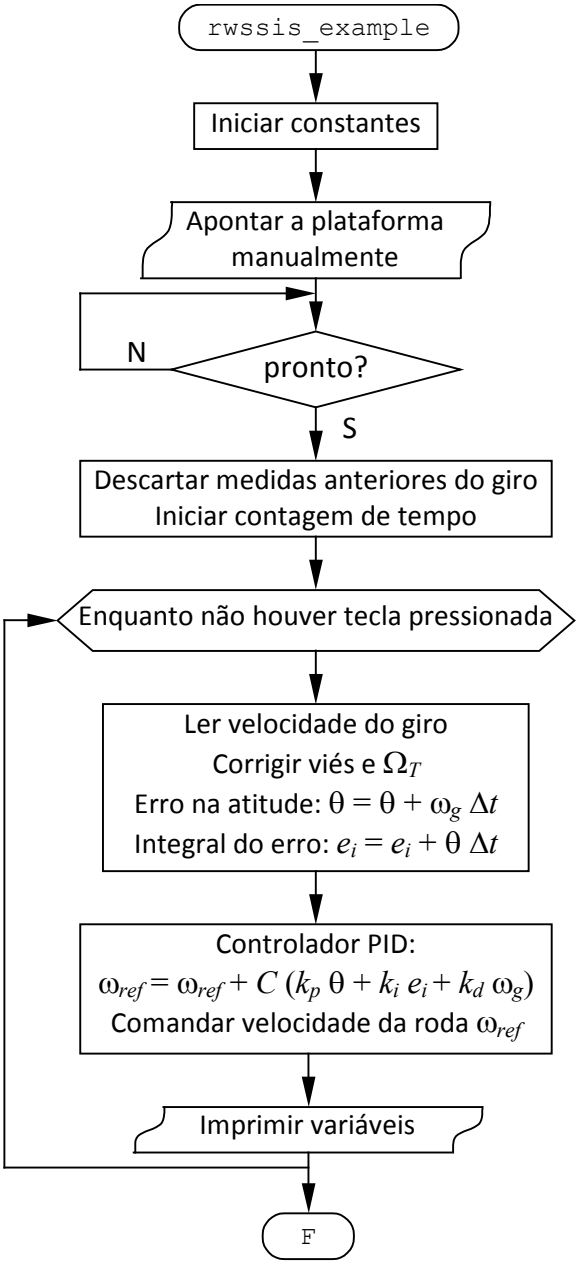

Fig. **13** – Fluxograma simplificado da função `rwssis_example` para controle de atitude da plataforma, com modo de operação em velocidade angular da roda.

O programa `Exemplo.cpp`, bem como todos os demais arquivos necessários para executá-lo encontra-se disponibilizado no sítio:

http://www2.dem.inpe.br/val/projetos/rwexp/SAABT.zip.

APÊNDICE A - REGISTRO E HISTÓRICO DO DOCUMENTO

Título:	Manual de operação da mesa de mancal a ar de um eixo	**ARQUIVO:** **DATA DE CRIAÇÃO: 28-ago-13** **PÁGINAS: 34** **REVISÃO: 2** **EDITOR: Microsoft Word 9.0**

Autoria	**PCT**	**Organização**	**Data**
Valdemir Carrara	SIA-SISCAO	INPE	28.8.2013
Aprovação			

Lista de Itens a serem Definidos ou Confirmados

Item	Descrição	Estado