Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica

**Rafael Guedes Lang**

# Desenvolvimento de Biblioteca de Sensoriamento e Comunicação para Sistemas Móveis Autônomos Microcontrolados

São Carlos
2012

**Rafael Guedes Lang**

# Desenvolvimento de Biblioteca de Sensoriamento e Comunicação para Sistemas Móveis Autônomos Microcontrolados

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos
da Universidade de São Paulo

Curso de Engenharia de Computação com ênfase
em Sistemas Computacionais Avançados

Orientador: Prof. Dr. Eduardo Morgado Belo

São Carlos

2012

Ficha catalográfica preparada pela Seção de Atendimentos ao Usuário do Serviço de Biblioteca – EESC/USP

L271d

Lang, Rafael Guedes.
Desenvolvimento de biblioteca de sensoriamento e comunicação para sistemas móveis autônomos microcontrolados. / Rafael Guedes Lang; orientador Eduardo Morgado Belo. –- São Carlos, 2012.

Monografia (Graduação em Engenharia da Computação com ênfase em Sistemas Computacionais Avançados) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012.

1. Sistemas móveis autônomos. 2. Redes de sensores. 3. Redes de microcontroladores. I. Titulo.

# FOLHA DE APROVAÇÃO

Nome: Rafael Guedes Lang

Título: "Desenvolvimento de Biblioteca Microcontrolada Distribuída de Sensoriamento e Comunicação para Aplicação em Sistemas Móveis Autônomos"

Trabalho de Conclusão de Curso defendido e aprovado em 07 / 02 / 2012,

com NOTA 7,0 (sete, zero), pela comissão julgadora:

Prof. Associado Evandro Luís Linhari Rodrigues - SEL/EESC/USP

Prof. Associado Ivan Nunes da Silva - SEL/EESC/USP

*Prof. Associado Evandro Luís Linhari Rodrigues*
*Coordenador pela EESC/USP do*
*Curso de Engenharia de Computação*

À minha família e aos meus amigos,
cuja companhia por vezes tive que
abrir mão para completar meus projetos.
Obrigado pelo apoio e pela compreensão!

Ao parceiro Smile,
que, como sempre,
deve parar sua leitura neste ponto.

“Dizem que sou louco por pensar assim
Se eu sou muito louco por eu ser feliz
(...)
Sim sou muito louco, não vou me curar
Já não sou o único que encontrou a paz
Mas louco é quem me diz e não é feliz,
eu sou feliz!!”

**Balada do Louco**
**Arnaldo Baptista e Rita Lee, 1986**

# Resumo

Este trabalho apresenta uma biblioteca capaz de gerenciar uma rede de microcontroladores para sensoriamento de sistemas móveis autônomos. Desde o seu surgimento, a complexidade, difusão e habilidade desses sistemas vêm aumentando constante e rapidamente. A necessidade de se localizar e locomover tornou-se tão complexa que alguns sistemas precisam de dados de vários sensores e a uma taxa de atualização muito alta, como o caso de um foguete de pequeno porte, que tem movimentos e reações rápidas e agressivas. Soluções com sensores heterogêneos têm sido propostas na literatura, no entanto poucas conseguem capturar e processar dados com velocidades realmente elevadas. Neste trabalho é proposta uma biblioteca que permite a utilização de vários microcontroladores em rede para distribuir o processamento e permitir maiores taxas e variedades de sensores. A implementação traz pronta toda a parte de comunicação entre os módulos utilizados e o funcionamento de sistemas de comando, telemetria e armazenamento, deixando para o desenvolvedor apenas a tarefa de implementar a captura e configuração dos sensores que serão utilizados. A biblioteca foi submetida a testes para verificar o funcionamento dos módulos e o desempenho em relação à leitura e ao processamento de dados. Todos os módulos funcionaram corretamente e a velocidade de leitura foi superior duas vezes e meia à melhor proposta apresentada na literatura. O alto desempenho e a modularização da biblioteca fazem-na uma ótima solução para desenvolvimento de sistemas de navegação, evitando o trabalho de baixo nível e permitindo um melhor foco nos pontos mais importantes do sistema.

Palavras-chave: Sistemas móveis autônomos. Redes de sensores. Redes de microcontroladores.

# Abstract

This work presents a library capable of managing a microcontrollers network for sensing of autonomous mobile systems. From its early times, the complexity, diffusion and ability of theses systems has been improving constantly. The need to localization and locomotion has become so complex that some systems require data from several sensors at high update rates, as a small rocket, for example. Solutions with heterogeneous sensors have been presented in lectures, however only a few can achieve really high speed data sensing and processing. In this work is introduced a library that allows the use of many microcontrollers in network in order to distribute the processing and reach better update rates and number of sensors. The implementation features all internal communication as well as systems for command, telemetry and storage, leaving only the implementation of sensor capturing and configuring for the developer. The library was tested to verify the functioning of its modules and the performance when reading and processing data. All modules worked fine and the reading rate was two and a half time faster than the best presented proposal. The library's high performance together with its modularization turn it in a great solution for the development of navigation systems, avoiding low level jobs and focusing in more important parts of the system.



Keywords: Autonomous mobile systems. Sensors network. Microcontrollers network.

# Lista de ilustrações

# Lista de códigos-fonte

# Lista de tabelas

# Lista de abreviações e siglas

| | |
|---|---|
| *ACK* | Confirmação (em inglês, *acknowledgement*) |
| *ADC* | Conversor analógico-digital (em inglês, *analog-to-digital converter*) |
| *ECAN* | Rede aprimorada de área de controlador (em inglês, *enhanced controller area network*) |
| *GCC* | *GNU Compiler Collection* |
| *GPS* | Sistema de posicionamento global (em inglês, *global positioning system*) |
| $I^2C$ | Inter-circuitos integrados (em inglês, *inter-integrated circuits*) |
| *MEMS* | Sensores microeletromecânicos (em inglês, *microelectromechanical systems*) |
| PDS | Processamento digital de sinais |
| *PIG* | Robô inspetor de tubulação (em inglês, *pipeline inspection gauge*) |
| *PWM* | Modulação por largura de pulso (em inglês, *pulse width modulation*) |
| *SPI* | Interface serial de periféricos (em inglês, s*erial peripheral interface*) |
| *UART* | Receptor transmissor assíncrono universal (em inglês, *universal asynchronous receiver transmiter*) |
| *UAV* | Veículo aéreo não-tripulado (em inglês, *unmanned air vehicle*) |

# Lista de símbolos

| | |
|---|---|
| bit/s | Bits por segundo, unidade de taxa de transferência |
| g | Unidade de aceleração equivalente a 9,8066 $m/s^2$ |
| KHz | Quilohertz, unidade de frequência ($10^3$ Hertz) |
| Km | Quilometro, unidade de comprimento ($10^3$ metros) |
| Hz | Hertz, unidade de frequência |
| m | Metro, unidade de comprimento |
| mg | Mili-g, unidade de aceleração equivalente a $10^{-3}$ g |
| min | Minuto, unidade de tempo |
| MHz | Megahertz, unidade de frequência ($10^6$ Hertz) |
| MIPS | Milhões de instruções por segundo, unidade de capacidade de processamento |
| W | Watt, unidade de potência |

# Sumário

# 1 Introdução

Sistemas móveis autônomos são sistemas capazes de se localizar e locomover de forma a realizar uma tarefa sem a intervenção humana. Durante a segunda guerra surgiram as bombas inteligentes e os foguetes alemães *V1* e *V2*, capazes de voar com um piloto automático primitivo e de detonar a uma distância definida do alvo. Desde então os sistemas evoluíram de forma a se tornar mais fortes, rápidos, robustos e úteis. Além desta evolução técnica, os sistemas autônomos se difundiram, estando presentes hoje em inúmeros segmentos do conhecimento e até no cotidiano de forma comercial. Há robôs agrícolas, como colheitadeiras e cortadores de grama; marítimos, como torpedos teleguiados; aéreos, como mísseis e veículos aéreos não-tripulados (em inglês, *UAVs*); de uso doméstico, como o aspirador de pó autônomo *Roomba* e o limpador de piscinas *Verro*, ambos da *iRobot*; de entretenimento, como o cachorro *Aibo*, da *Sony*; e até robôs que jogam futebol, como os participantes da *RoboCup*[1].

Embora sejam tarefas bastante diferentes, todos os sistemas têm em comum a necessidade de se localizar e locomover para realizá-las, ações conhecidas como navegação e guiagem. Navegação é a habilidade do sistema de obter dados de sensores e até informações de outros sistemas para saber aonde ele se localiza no espaço e qual é a sua orientação em relação a um referencial definido. Guiagem é a ação de se manter em uma trajetória definida de locomoção, evitando deixá-la e, consequentemente, errar seu destino.

Por serem autônomos e móveis, é comum esses sistemas serem alimentados por baterias, o que torna seu consumo de potência um problema e o uso de microcontroladores para a captura e interpretação dos sensores uma solução, uma vez que eles têm baixo consumo e tamanho reduzido. A crescente diversidade de objetivos e ambientes de atuação dos sistemas móveis traz a necessidade de captura, interpretação e processamento de diferentes tipos e grandes quantidades de sensores para uma navegação mais rápida, precisa e confiável.

Emmendoerfer (2008) e Santana (2011) propõe sistemas de navegação

1 Competição mundial na qual robôs autônomos disputam partidas de futebol

para robôs inspetores de tubulação (em inglês, *PIG*). O primeiro apresenta um sistema elaborado com componentes caros e de alta precisão, permitindo leituras em frequências de até 1600 Hz. São utilizados três acelerômetros, três giroscópios baseados em *laser* e um receptor de *Global Positioning System* (*GPS*), além de sistemas auxiliares como medidores de temperatura e hodômetros. Já o segundo apresenta um sistema comercial mais simples e barato, com taxa de atualização de 100 Hz, mas com uma implementação de filtros de *Kalman* para melhoria de precisão, alcançando erros de apenas 3 m em 15 min ou 2,7 Km de operação. Neris (2001), França Jr. (2009) e Lopes (2010) trazem modelagens e simulações para sistemas de navegação de aeronaves autônomas, os primeiros com aviões e a última com helicópteros. Lavieri (2011), em sua dissertação de mestrado, desenvolve um sistema para navegação de veículos submarinos, particularmente as estacas-torpedo[2], com sensores comerciais e capaz de funcionar também a 100 Hz. Paixão (2010) implementa um sistema próprio para a aeronave não tripulada do Instituto Militar de Engenharia, com acelerômetros, giroscópios e bússola de baixo custo, obtendo taxas de leitura de 50 Hz.

Sistemas de alta velocidade e aceleração, como robôs de exploração e foguetes de pequeno porte, necessitam, além de vários tipos de sensores, de alta frequência de atualização em sua navegação, uma vez que suas forças e respostas são intensas, proporcionando rápidas mudanças de posição e orientação. Das implementações apresentadas, apenas Emmendoerfer(2008) traz altas taxas de atualização, possíveis em um sistema comum apenas com componentes de alto custo.

O objetivo deste trabalho é apresentar uma biblioteca genérica para redes de microcontroladores que ofereça suporte a essas altas taxas de captura, processamento e armazenamento de informações, proporcione módulos de comando e telemetria, permita reconfiguração *on-the-fly* e seja de fácil utilização e manutenção por parte do desenvolvedor.

2 Veículo submarino para ancoragem de plataformas de petróleo

# 2 Revisão teórica

Nos tópicos que seguem são apresentadas algumas tecnologias interessantes para um correto entendimento deste trabalho, uma vez que grande parte deles é utilizada na base da biblioteca nele proposta.

## 2.1 Microcontroladores

Microcontroladores são circuitos integrados que contém os elementos básicos de um computador: unidade de processamento, memória e dispositivos de entrada e saída. Sua principal aplicação é em sistemas embarcados, uma vez que são pequenos e programáveis, permitindo a integração com diversos sensores e atuadores eletrônicos. São caracterizados principalmente pelo seu consumo, frequência de operação, largura de barramento e número de periféricos e portas. Há modelos simples, com poucas portas e baixa velocidade, voltados para aplicações de baixíssimo consumo, bem como modelos maiores e mais potentes com capacidade de se comunicar por inúmeros barramentos, capturar vários sinais analógicos simultaneamente e realizar tarefas tão complexas quanto controle de plantas e processamento de imagens em tempo real.

### 2.1.1 Família dsPIC33F

A família de microcontroladores *dsPIC33F*, da estadunidense *Microchip Technology*, é formada por dispositivos de arquitetura *Harvard* modificada, com barramento de dados de 16 bits e instruções adicionais de processamento digital de sinais (PDS). Há modelos de 28 a 100 pinos, com frequência de operação entre 32 e 120 MHz, oferecendo diversos periféricos como até vinte e quatro conversores analógico-digitais (em inglês, *ADC*) de 12 bits, duas interfaces *Serial Peripheral Interface* (*SPI,* em português, Interface Serial de Periféricos), quatro módulos *Universal Asynchronous Receiver Transmiter* (*UART,* em português, Receptor Transmissor Assíncrono Universal) e quatro geradores independentes de *Pulse Width Modulation* (*PWM*, em português, Modulação por Largura de Pulso) (Microchip, 2011).

Além de conter um amplo conjunto de instruções de máquina, incluindo as de PDS, *dsPICs* podem ser programados em linguagem C utilizando o compilador C30, também da *Microchip Technology*, uma modificação do popular *GNU Compiler Collection* (*GCC*), tornando o desenvolvimento nessa plataforma bastante eficiente, rápido e confiável (Microchip, 2011).

O modelo *dsPIC33FJ256MC710*, mostrado na figura 1, é capaz de processar a 40 MIPS e contém cem pinos, sendo oitenta e cinco portas programáveis de entrada e saída. O dispositivo traz também vinte e quatro canais de conversão analógico-digital, dois módulos *Inter-Integrated Circuit* (*$I^2C$,* em português, Inter-Circuitos Integrados), duas interfaces *SPI*, dois módulos *UART*, duas interfaces *Enhanced Controller Area Network* (*ECAN,* em português, Rede Aprimorada de Área de Controlador) e cinco pinos de interrupção externa (Microchip, 2009).

Figura 1 - Fotografia do microcontrolador dsPIC33FJ256MC710

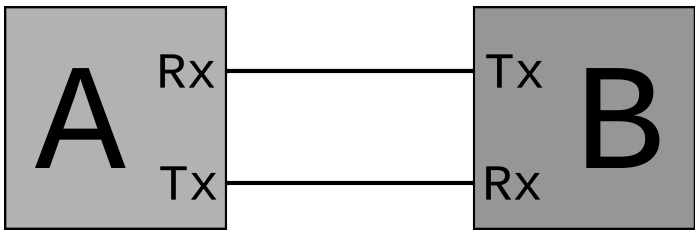

## 2.2 Barramentos de comunicação

Em sistemas digitais existem inúmeras possibilidades de métodos e protocolos de comunicação. Há sempre um compromisso entre taxa de transmissão, número de linhas e complexidade de implementação, o que levou à padronização de algumas abordagens mais eficientes. Três dessas alternativas são apresentadas a seguir, pois são utilizadas neste trabalho.

### 2.2.1 “Universal Asynchronous Receiver Transmiter”

*UARTs* são sistemas *full-duplex*[3] nos quais os dados são transmitidos serialmente por duas linhas principais: entrada (*Rx*) e saída (*Tx*). Não há linha de *clock*, portanto a comunicação é assíncrona. Devido a isso é necessário que ambas as partes da comunicação saibam previamente a taxa e o formato de envio dos dados. Na figura 2 está representada a conexão básica para uma comunicação utilizando esse protocolo.

Figura 2 - Conexão básica para comunicação por “UART”




A informação é transmitida em pacotes: a comunicação é iniciada com um nível lógico 0 (*Start Bit*), em seguida são enviados sete ou oito bits com a informação desejada, seguidos por um opcional bit de paridade e finaliza-se enviando um ou dois níveis lógicos 1 (*Stop Bits*). Na figura 3 é mostrada uma das possíveis configurações desse pacote.

Figura 3 - Pacote típico de comunicação do “UART”

| Bit de Início | Dado 0 | Dado 1 | Dado 2 | Dado 3 | Dado 4 | Dado 5 | Dado 6 | Dado 7 | Bit de Término |
|---|---|---|---|---|---|---|---|---|---|

Configuração do pacote: 8 bits de dado, sem bit de paridade e 1 *stop bit*

### 2.2.2 “Inter-Integrated Circuit”

O *I²C* é um barramento síncrono, de comunicação serial, composto por duas linhas: *clock* (*SCL*) e dados (*SDA*), proposto em 1982 pela *NXP*

3 Sistemas *full-duplex* têm a capacidade de enviar e receber dados simultaneamente. Já os *half-duplex* conseguem executar apenas uma dessas operações de cada vez.

*Semiconductors*, antiga divisão de semicondutores da *Royal Philips Electronics*. Seu funcionamento é *half-duplex*[3], baseado em um sistema mestre-escravo com endereços, havendo a possibilidade de vários mestres e escravos (Paret, 1997). Todos os dispositivos são capazes de enviar dados, no entanto apenas um, o mestre, pode gerar o *clock* da comunicação. A transmissão é iniciada pelo mestre, que gera uma condição de *Start* (nível lógico 1 em *SCL* e transição em *SDA* de 1 para 0) e envia um pacote contendo o endereço do escravo que deseja acessar. O escravo responde com um estado de confirmação (*ACK*, do inglês *acknowledgement*) e a partir desse momento os dados são transmitidos, também com sinais de *ACK*, do mestre para o escravo e vice-versa de acordo com a necessidade da "conversa", que é finalizada com uma condição de *Stop* (nível lógico 1 em *SCL* e transição em *SDA* de 0 para 1).

### 2.2.3 "Serial Peripheral Interface"

A *SPI* é um protocolo síncrono do tipo mestre-escravo de três ou quatro linhas elaborado pela *Motorola, Inc*. Embora seja limitada a apenas um mestre, permite a comunicação com vários escravos, que são selecionados pela linha *chip select* ($\overline{CS}$, em português, seleção de dispositivo).

Além da $\overline{CS}$, no padrão *full-duplex* de quatro linhas são definidas as linhas únicas direcionais de *clock* (*SCLK*), entrada do mestre/saída do escravo (*MISO*) e saída do mestre/entrada do escravo (*MOSI*). Na variação de três linhas, além das linhas $\overline{CS}$ e *SCLK*, há apenas uma linha compartilhada de dados, tornando o sistema *half-duplex*. Em qualquer um dos padrões, é conectado uma porta de saída do mestre à linha $\overline{CS}$ de cada escravo, conforme mostra a figura 4. A arquitetura básica de uma *SPI* é composta pelas quatro linhas, um gerador de *clock* (no caso do mestre) e dois registradores, um de entrada e um de saída. Esses elementos são mostrados na figura 5, adaptada de Microchip (2009), que contém um esquemático da *SPI* dos microcontroladores da família *dsPIC33F*.

Figura 4 - Configuração de dispositivos para “SPI” com vários escravos

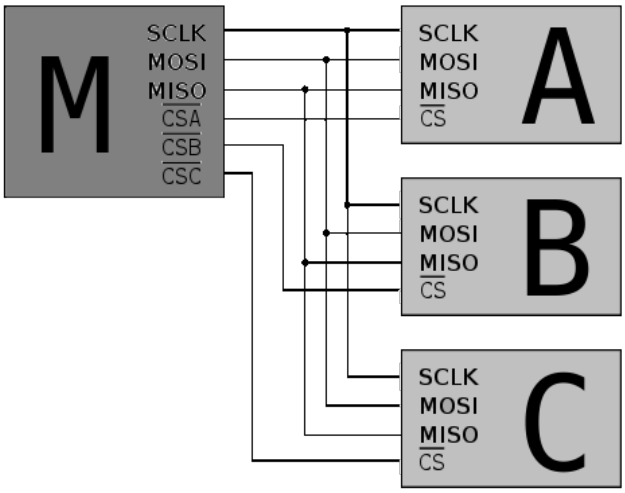
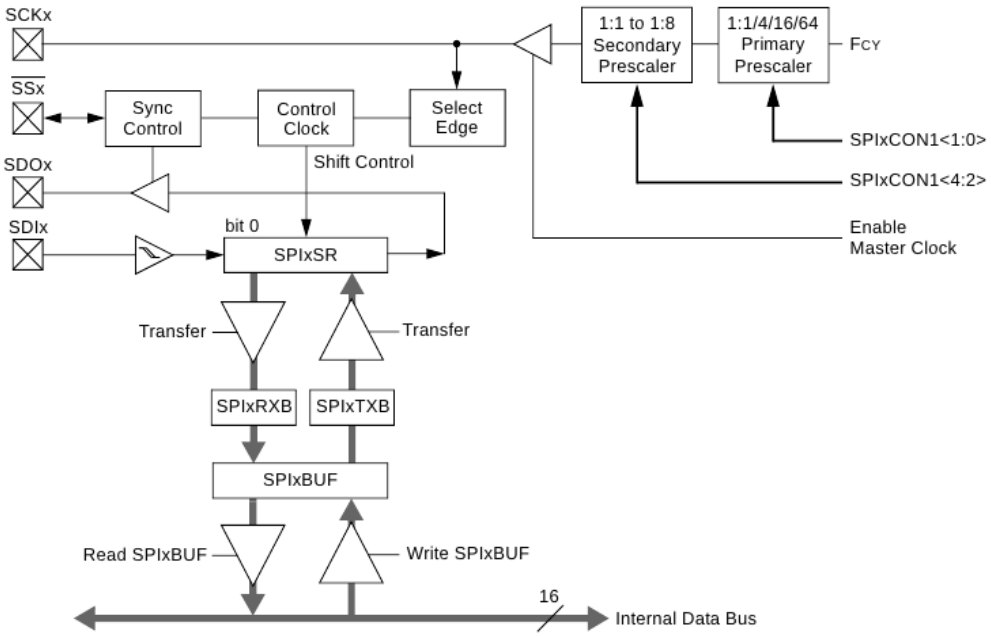



Figura 5 - Arquitetura do “SPI” implementado na família “dsPIC33F”



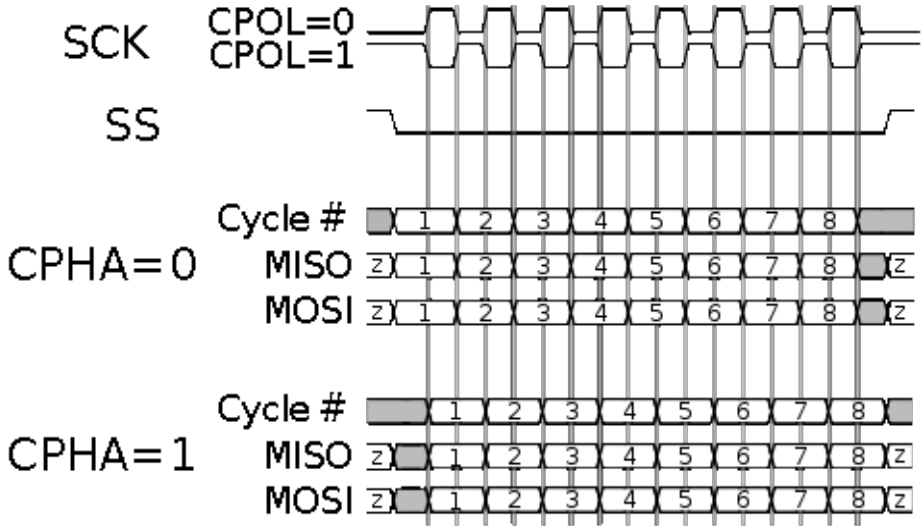

**Fonte:** Microchip, 2009

A transmissão é iniciada pelo mestre, que atribui nível lógico 0 à linha $\overline{CS}$ do escravo com o qual deseja se comunicar e 1 às dos outros e em seguida começa a gerar o *clock* da comunicação na linha *SCLK*. A cada pulso de *clock* um bit é enviado pela linha *MOSI* do registrador de saída do mestre para o registrador de entrada do escravo. Simultaneamente, de forma análoga, o escravo envia dados ao

mestre pela linha *MISO*. Ao final da transmissão é cessado o *clock* e a linha $\overline{CS}$ é colocada novamente em nível lógico 1.

Como proposto em Motorola (2003) e exemplificado na figura 6, há quatro formas de polarização (*CPOL*) e fase (*CPHA*) do *clock* da comunicação. A polarização pode ser 0 ou 1, indicando o estado de base da linha *SCLK*, nível lógico no qual ela permanece quando não há comunicação. Já *CPHA* indica o momento da leitura do dado daquele pulso, sendo 0 quando a leitura é feita na saída do *clock* do estado de base e 1 quando é feita na chegada a tal estado.

Figura 6 - Modos de polarização e fase do “SPI”




**Fonte:** Motorola, 2003

## 2.3 Sensores

Sensores são dispositivos capazes de aferir uma grandeza física e transformá-la em grandezas elétricas. Existem sensores para inúmeras medições, como temperatura, aceleração, cor. Podem ser analógicos ou digitais. Os analógicos geram uma saída elétrica proporcional à grandeza medida, como, por exemplo, uma tensão de saída proporcional à quantidade de luz incidente no sensor. Já os digitais são sensores analógicos com uma eletrônica auxiliar, que converte as medições feitas para valores digitais e as armazena e transmite quando requisitado.

Há sensores que utilizam propriedades eletrônicas e químicas dos materiais para ler alguma informação. É o caso, por exemplo, dos sensores de temperatura e umidade do ar. O primeiro é formado por dispositivos semicondutores em uma configuração que amplifica sua sensibilidade à temperatura (Analog Devices, 1996). Já o segundo usa das propriedades químicas de certos elementos para gerar uma saída proporcional à umidade relativa do ar (Honeywell, 2010).

O surgimento dos chamados Sensores Microeletromecânicos (em inglês, *MEMS*) trouxe grande revolução à indústria de sensores. São dispositivos fabricados por processos semelhantes aos utilizados em dispositivos microeletrônicos, como deposição de camadas, *etching* e aplicação de bombardeamento por feixes de elétrons. Embora sua precisão seja consideravelmente menor que a dos sensores tradicionais, a integração de diversos dispositivos desse tipo pode trazer dados bastante confiáveis de forma barata, leve e pequena, uma vez que a área desses sensores já encapsulados é da ordem de poucos milímetros quadrados e seu preço é bastante menor que o dos sensores tradicionais (Analog Devices, 2004), (Analog Devices, 2009), (Bosch, 2009) e (Freescale, 2011).

Existem *MEMS* para diversos tipos de medições, como aceleração, velocidade angular e pressão. Alguns capazes de realizar medições em várias dimensões, como o caso de acelerômetros que medem a componente de aceleração em cada um dos três eixos cartesianos, e outros capazes de ter sua escala ajustada por parâmetros de configuração, possibilitando maior precisão quando possível. A utilização de um mesmo *ADC* para ler dois sensores diferentes, um com uma faixa duas vezes maior que a do outro, gera uma precisão duas vezes menor na leitura do primeiro sensor. Esse fato obriga a criação de sensores com escalas diferentes para que a medição possa ser a mais precisa possível dentro do intervalo requerido.

### 2.3.1 Acelerômetro “ADXL 345”

O acelerômetro digital *ADXL 345* é fabricado pela empresa *Analog Devices* e é capaz de medir aceleração nos três eixos cartesianos com escala e frequência ajustáveis, conforme mostra a tabela 1. Sua comunicação é feita por *SPI*

e sua configuração é baseada em registradores que podem ter seus valores alterados para ativar funções ou definir a frequência de operação, por exemplo.

Tabela 1 - Características do acelerômetro ADXL 345

| **Característica** | **Descrição** |
|---|---|
| Número de eixos | 3 |
| Tipo de saída | Digital (*SPI*) |
| Escala | Ajustável (±2, ±4, ±8, ±16 g) |
| Frequência de operação | Ajustável (0,1 a 3.200 Hz) |

**Fonte:** Adaptada de Analog Devices (2009)

## 2.4 Módulos de rádio “XTend 900”

Fabricado pela companhia *MaxStream*, o módulo de rádio *XTend 900* é um dispositivo de médio alcance capaz de realizar transmissões de 115.200 bits/s a 900 MHz com 1 W de potência.

Seu funcionamento é minimalista, sendo que o pacote recebido no *UART* é transmitido sem modificações para o destino, o que simula uma comunicação *UART* entre dois dispositivos em uma mesma placa, por exemplo.

Seu pacote de desenvolvimento traz um módulo para comunicação com computadores e um software para receber os dados, possibilitando armazenamento em disco para análise posterior ou utilização por outros programas para tratamento em tempo real.

# 3 Proposta

Visando cumprir os requisitos de alta taxa de aquisição, diversidade de tipos de sensores, processamento e reconfiguração *on-the-fly*, armazenamento e telemetria, este trabalho propõe uma biblioteca para microcontroladores composta pelos módulos e interfaces detalhados nas seções seguintes.

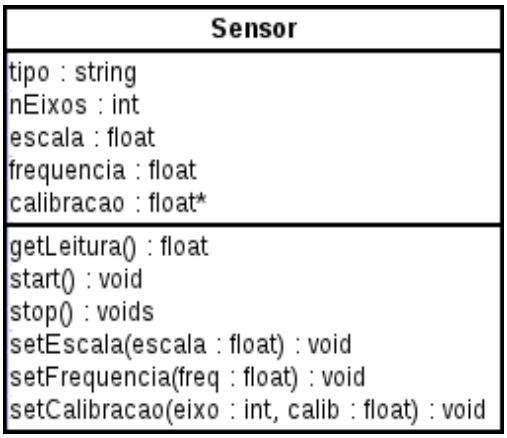
A biblioteca implementa toda a comunicação inter-módulos, o sistema de armazenamento e os métodos de publicação e descoberta dos módulos implementados, deixando a cargo do desenvolvedor apenas a parte de leitura e configuração dos sensores e implementação de algoritmos de processamento.

A maioria dos elementos pode rodar em qualquer dispositivo da rede, bastando apenas informar ao módulo superior em qual microcontrolador ele se encontra. O código foi desenvolvido para ser utilizado com a família *dsPIC33F* de microcontroladores, sendo que muitos dos métodos implementados utilizam módulos desses dispositivos. A utilização de uma rede de microcontroladores e a distribuição de processamento entre eles visa permitir a captura de mais sensores ao mesmo tempo.

## 3.1 Sensor

O sensor é a interface central da arquitetura e define protótipos para métodos de configuração, calibração e obtenção de dados, conforme mostrado na figura 7. A implementação da interface recebe comandos para se configurar ou retornar a medição feita e deve responder conforme o formato padronizado. O sensor tem tipo definido, escala e frequência de operação únicas, porém ajustáveis, e pode ter vários eixos e fatores de calibração para cada um deles. O desenvolvedor deve implementar todos os métodos definidos.

Figura 7 - Diagrama com atributos e métodos da interface Sensor




### 3.1.1 Descrição dos métodos

- *setEscala*: Método que deve configurar o sensor para que realize as leituras na escala escolhida;
- *setFrequencia*: Método que deve configurar o sensor para que realize as leituras na frequência escolhida;
- *setCalibracao*: Método para armazenar parâmetros de calibração que podem ser utilizados no método *getLeitura* para corrigir eventuais desvios do sensor;
- *start*/*stop*: Métodos para iniciar e parar a leitura de dados do sensor;
- *getLeitura*: Método principal da interface, deve realizar a leitura do dado do eixo pedido do sensor, convertê-lo para um número real dentro da escala, realizar os procedimentos de correção da calibração e retornar o valor resultante.

No código 1 é apresentado um exemplo de implementação do método *getLeitura*.

Código 1 - Exemplo de código-fonte em C para implementação do método “getLeitura” da interface Sensor

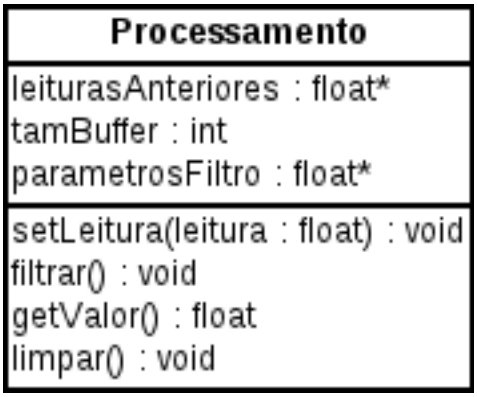
```
float sensor_getLeitura(Sensor* sensor, int eixo) {
    /* Declaracao de variaveis */
    int   dadoCru;
```

```
    float valor;

    /* Le o dado do sensor */
    dadoCru = lerDado(eixo); // Funcao especifica do sensor

    /* Aplica calibracao e escala */
    valor  = dadoCru * sensor.escala / sensor.resolucao;
    valor += sensor.calibracao[eixo];

    /* Retorna o valor em formato */
    return(valor);
}
```

## 3.2 Processamento

A interface de processamento define métodos para filtragem das leituras feitas e pode não ser utilizada caso se queira apenas os dados não-filtrados. Na figura 8 estão os métodos por ela definidos. Os métodos de gerenciamento do vetor com leituras anteriores são implementados pela biblioteca, restando ao desenvolvedor implementar os métodos *filtrar* e *getValor*.

Figura 8 - Diagrama com atributos e métodos da interface Processamento




### 3.2.1 Descrição dos métodos

- *setLeitura*: Método que insere um valor lido no vetor de leituras anteriores. O vetor é preenchido removendo a leitura mais antiga e adicionando a nova;

- *limpar*: Método que remove todas as entradas do vetor de leituras anteriores;
- *filtrar*: Método que deve, a partir dos dados contidos no vetor de leituras anteriores, realizar operações que gerem como resultado um número que represente de forma geral os dados contido em tal vetor;
- *getValor*: Método que retorna o último valor calculado pelo método *filtrar*.

O armazenamento dos últimos valores lidos permite a implementação de uma grande gama de filtros digitais, como os estatísticos e os de *Kalman*. No código-fonte 2 é apresentada a implementação do método *filtrar* para um filtro de média.

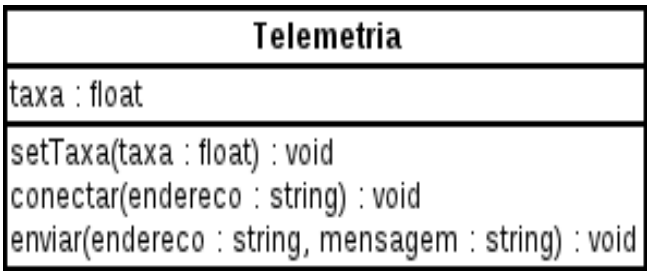
Código 2 - Exemplo de código-fonte em C para implementação do método “filtrar” da interface Processamento utilizando um filtro de média

```
void processamento_filtrar(Processamento* processamento) {
    /* Declaracao de variaveis */
    int   i;

    /* Zera a variavel de valor */
    processamento.valor = 0.0;

    /* Percorre o buffer calculando sua media */
    for (i = 0; i < processamento.tamBuffer; i++) {
        valor += leiturasAnteriores[i];
    }

    /* Calcula a media */
    valor /= processamento.tamBuffer;
}
```

## 3.3 Telemetria

O módulo de telemetria é implementado pela biblioteca como o encapsulamento de uma interface *UART*. Os dados recebidos pela função *enviar* são repetidos na linha *Tx* da *UART* na taxa definida pelo método *setTaxa*. Embora não esteja implementado, é definido um endereço de destino, que poderia ser tratado como um conjunto de saídas *UART* de forma que o módulo enviasse a mensagem à saída escolhida pelo endereço. Os métodos do módulo são apresentados na figura 9.

Figura 9 - Diagrama com atributos e métodos da interface Telemetria




O controlador *UART* utilizado é o disponível nos microcontroladores escolhidos. Sua configuração é feita atribuindo valores aos registradores do dispositivo que controlam tal módulo, o que permite uma rápida e confiável troca de taxa de comunicação. O envio dos dados também é feito em baixo nível, o que permite um melhor controle da operação e do formato dos dados enviados.

## 3.4 Armazenamento

Conjunto de métodos utilizados para armazenar os dados de forma padronizada em um cartão de memória *microSD* utilizando a biblioteca *FSIO* da *Microchip Technology*. A biblioteca do sistemas de arquivo, que utiliza o barramento *SPI*, é encapsulada e a ela é adicionado um gerenciador de arquivos existentes, que permite saber quais arquivos existem, além de criar novos, o que torna o armazenamento mais organizado. O desenvolvedor utiliza as funções de gerenciamento e o método principal de armazenamento, *armazenar*, no qual informa o arquivo destino, a origem e os dados a serem gravados. Na figura 10 são mostrados esses métodos em um diagrama.

Figura 10 - Diagrama com atributos e métodos da interface Armazenamento


O formato dos dados no arquivo gerado é apresentado na figura 11.

Figura 11 - Formato de entrada no arquivo de dados

```
tempo # código_do_sensor $ número_do_eixo % dado
```

As entradas são números inteiros sem sinal,
exceto “dado” que é um número real

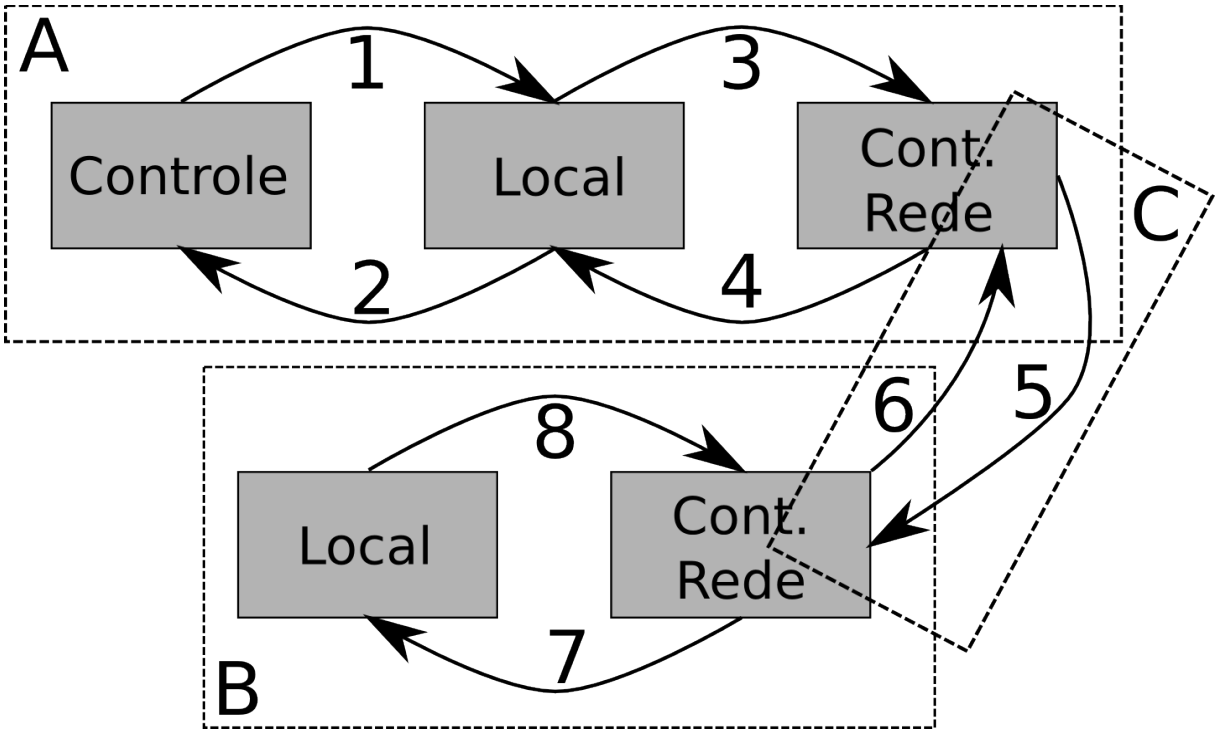
## 3.5 Módulo de controle e comando

São, ao mesmo tempo, os módulos mais importantes da biblioteca e os menos visíveis ao desenvolvedor. São responsáveis por gerenciar as instâncias das interfaces que existem no sistema bem como a comunicação entre elas na rede.

São formados por controladores locais e controladores de rede. Os controladores locais gerenciam a lógica dentro do seu próprio microcontrolador, chamando e passando parâmetros entre os métodos das instâncias que estão sendo executadas neste dispositivo. Quando uma instância local precisa se comunicar com outra que está em outro dispositivo, o controlador central conversa, utilizando o barramento *SPI*, com o controlador de rede desse outro microcontrolador, que recebe os comandos e os repassa ao controlador local daquele dispositivo para que ele possa incluí-los em suas ações.

Na figura 12 é apresentado um diagrama que mostra o fluxo das chamadas e dos dados em uma requisição do sistema.

Figura 12 - Diagrama de descoberta de módulos na rede




A – Módulos internos do microcontrolador requisitante
B – Módulos internos do microcontrolador requisitado
C – Barramento *SPI* de comunicação

Abaixo são exemplificadas as funções das arestas de 1 a 8 da figura 12 para uma chamada da função *getLeitura*:

- 1: O controle requisita a leitura de algum sensor ao controlador local. Caso o sensor esteja no mesmo microcontrolador, a função *getLeitura* é chamada e seu retorno é passado para a ação 2. Caso contrário, o controlador local realiza a ação 3;
- 2: Retorna o valor recebido;
- 3: Caso o sensor pedido não esteja no mesmo microcontrolador, o controlador local pede ao controlador de rede que repasse a requisição ao microcontrolador no qual está localizado o sensor (ação 5);
- 4: Retorno do valor recebido da rede;
- 5: O controlador de rede do microcontrolador de origem verifica em sua tabela em qual microcontrolador o sensor requisitado está e envia um comando

pedido a ele o retorno de uma chamada da função *getLeitura*;

- 6: Retorno pela rede do valor lido no sensor requisitado;
- 7: O controlador de rede requisitado chama a função *getLeitura* do sensor requisitado;
- 8: Retorno da função *getLeitura*.

# 4 Experimentos

Para realizar a validação e o teste da biblioteca foi implementado um sistema protótipo com três microcontroladores, um acelerômetro digital de três eixos e dois rádios para comando e telemetria.

A placa contendo os componentes foi fabricada no laboratório de *hardware* do grupo *Warthog Robotics* da USP São Carlos por processo de corrosão química e soldada parte em forno elétrico, parte à mão. Na tabela 2 são apresentados os componentes utilizados pelo sistema.

Tabela 2 - Modelos de dispositivos utilizados no sistema protótipo

| **Dispositivo** | **Fabricante** | **Modelo** |
|---|---|---|
| Microcontrolador | *Microchip Technology* | *dsPIC33FJ256MC710* |
| Acelerômetro | *Analog Devices* | *ADXL 345* |
| Rádio | *MaxStream* | *XTend 900* |

O protótipo foi submetido aos experimentos descritos nas próximas seções para que fosse possível uma avaliação do correto funcionamento e do desempenho da biblioteca.

## 4.1 Medição da gravidade com telemetria e reconfiguração

O primeiro experimento visou testar o funcionamento correto dos sistemas de comando, telemetria e captura de dados. Para isso, o sistema foi configurado com um acelerômetro ligado a um microcontrolador e os módulos de comando e telemetria a um segundo. O experimento consistiu em medir a aceleração devido à gravidade da Terra com o acelerômetro, que é capaz de ter sua escala alterada. A gravidade foi escolhida por ser um fenômeno praticamente invariante em pequenos intervalos de tempo dada uma altitude constante, o que permitiu uma validação fácil e precisa dos dados obtidos.

O sistema capturou dados a uma frequência de 50 Hz durante um intervalo de tempo, que foi divido em três regiões, entre as quais o módulo de

comando reconfigurou a escala do acelerômetro conforme a tabela 3. O módulo do vetor resultante das medições dos três eixos foi usado como dado, evitando assim possíveis problemas de alinhamento da montagem. Os dados obtidos foram enviados sem filtragem pelo rádio ligado ao módulo de telemetria a outro rádio conectado a um computador, que os armazenou.

Tabela 3 - Escala de captura do acelerômetro nas regiões do teste de aceleração da gravidade

| Região | Escala do acelerômetro (g) |
|---|---|
| A | ± 2 |
| B | ± 8 |
| C | ± 16 |

No código-fonte 3 é apresentada a função que implementa o método de leitura dos dados do acelerômetro.

Código 3 - Código-fonte em C da função de leitura dos dados do acelerômetro “ADXL 345”

```
float sensor_getLeitura(Sensor* sensor, int eixo) {
    /* Declaracao de variaveis */
    int   comandoA, comandoB, dadoCru;
    float valor;

    /* Atribui o valor do comando de acordo com o eixo */
    switch (eixo) {
        /* Eixo x */
        case 0: {
            comandoA = 0xB200;
            comandoB = 0xB300;
            break;
        }
        /* Eixo y */
        case 1: {
            comandoA = 0xB400;
            comandoB = 0xB500;
            break;
        }
        /* Eixo z */
        case 2: {
            comandoA = 0xB600;
            comandoB = 0xB700;
            break;
        }
    }

    /* Atribui 0 a porta CS do acelerometro */
    ADXL345_CS = 0;

    /* Envia o comando de leitura dos
       primeiros oito bits de dados */
```

```
    WriteSPI1(comandoA);
    while (!SPI1_TX_DONE) {}

    /* Le o dado */
    dadoCru = ReadSPI1() & 0xFF;

    /* Envia o comando de leitura dos
       outros oito bits de dados */
    WriteSPI1(comandoB);
    while (!SPI1_TX_DONE) {}

    /* Le o dado */
    dadoCru |= (ReadSPI1() & 0xFF) << 8;

    /* Atribui 1 a porta CS do acelerometro */
    ADXL345_CS = 1;

    /* Faz a conversao para a escala atual */
    valor  = dadoCru * sensor.escala / sensor.resolucao;
    valor += sensor.calibracao[eixo];

    /* Retorna o valor lido */
    return(valor);
}
```

## 4.2 Vários sensores com filtro e armazenamento local

O segundo experimento verificou a capacidade de captura, processamento e armazenamento de dados de vários sensores.

A interface de sensor foi implementada por um pseudo-sensor, definido no código 4. Cada pseudo-sensor foi configurado com um valor-base e gerou saídas com ruído branco gaussiano em torno desse valor. A média da curva gerada por um pseudo-sensor converge para o valor-base, o que permitiu uma forma fácil de testar o sistema de processamento, que foi implementado com um filtro que calcula a média das últimos 50 leituras.

No experimento foram utilizados dois microcontroladores para captura, cada um alocando dez pseudo-sensores e suas implementações de filtro de média, além de um microcontrolador para o armazenamento dos dados.

Código 4 - Código-fonte em C da função de geração de valores aleatórios do pseudo-sensor

```
#define MAX_VARIACAO 0.1

float sensor_getLeitura(Sensor* sensor, int eixo) {
    /* Declaracao de variaveis */
    float valor, variacao;

    /* Utiliza o parametro de calibracao do sensor
       como base do valor a ser gerado */
    valor = sensor.calibracao[0];

    /* Calcula a variacao */
    variacao = rand() % valor*MAX_VARIACAO;

    /* Decide o sinal da variacao */
    if (rand() % 2 == 0) {
        variacao *= -1.0;
    }

    /* Soma a variacao ao valor-base */
    valor += variacao;

    /* Retorna o valor gerado */
    return(valor);
}
```

Os dados foram gerados a uma frequência de 4 Khz e o armazenamento foi feito a 400 Hz. Para validar os dados obtidos, como variáveis de controle, todos os dados gerados por dois dos pseudo-sensores foram armazenados em um cartão de memória separado.

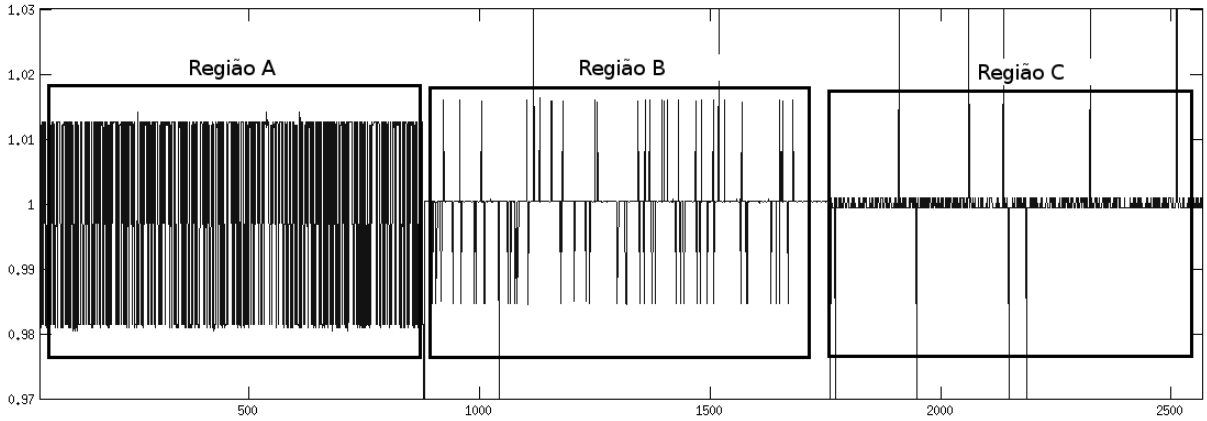
# 5 Resultados e discussão

## 5.1 Medição da gravidade com telemetria e reconfiguração

No experimento de medição da aceleração da gravidade todos os módulos funcionaram corretamente. Na figura 13 é apresentada a curva lida no sistema de telemetria. Nela vê-se três regiões bastante distintas, anotadas como A, B e C, conforme a tabela 3. Na primeira, que representa o acelerômetro funcionando em ±2 g, percebe-se que os valores oscilam bastante em torno do valor-base de 1 g. Na segunda essa oscilação diminui e na última é praticamente inexistente. Esse fato, mostrado em números na tabela 4, comprova o funcionamento de todas as partes testadas nesse experimento, uma vez que:

- Os dados foram obtidos no computador de telemetria e com os valores corretos, o que mostra que os sistemas de telemetria, captura e a comunicação inter-módulos funcionam;
- O sistema de comando funciona, já que a escala do sensor foi corretamente alterada durante a execução do experimento, gerando o resultado esperado: a redução do desvio padrão ao aumentar a escala do sensor, mantendo a média.

Figura 13 - Curva temporal do módulo do vetor de aceleração versus número da medição obtida pelo sensor “ADXL 345”




As marcações indicam as regiões relacionadas na tabela 3
Curva gerada sem filtragem dos dados

Tabela 4 - Média e desvio padrão das medições do módulo de aceleração da gravidade em cada região de operação no experimento da aceleração gravitacional

| Região | Média (g) | Desvio padrão (mg) |
|---|---|---|
| A | 1,0 | 15,5 |
| B | 1,0 | 5,1 |
| C | 1,0 | 3,6 |

## 5.2 Vários sensores com filtro e armazenamento local

Neste segundo experimento pode-se, novamente, verificar o funcionamento correto da biblioteca. As figuras 14 e 15 mostram os valores gerados pelos pseudo-sensores usados como controle, um com valor-base -1 e outro com 2. A tabela 5 mostra dados de média e desvio dessas curvas, comprovando a geração correta dos números aleatórios com variação limitada. Nas figuras 16 e 17 são mostrados os sinais salvos pelo módulo de armazenamento após a filtragem e na tabela 6 há dados desses valores.

Figura 14 - Curva de valor gerado pelo pseudo-sensor de valor-base -1 versus número da medição

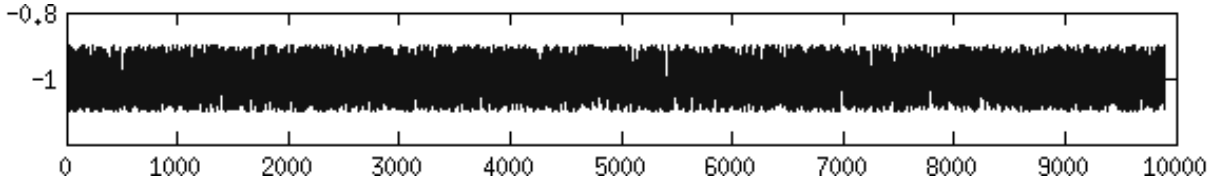



Figura 15 - Curva de valor gerado pelo pseudo-sensor de valor-base 2 versus número da medição

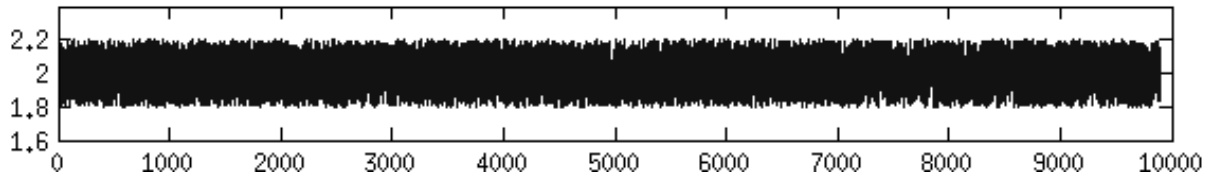


Tabela 5 - Dados estatísticos das curvas originais geradas pelos pseudo-sensores

| Valor-base | Média | Desvio padrão |
|---|---|---|
| -1,0 | -0,999 | 0,058 |
| 2,0 | 2,000 | 0,116 |

Tabela 6 - Dados estatísticos das curvas armazenadas em cartão após filtragem

| Valor-base | Média | Desvio padrão |
|---|---|---|
| -1,0 | -0,999 | 0,008 |
| 2,0 | 2,000 | 0,016 |

Figura 16 - Curva de valor filtrado a partir de sensor de valor-base -1 versus número da medição

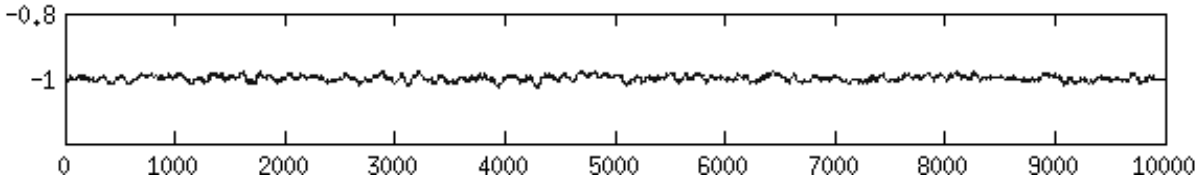
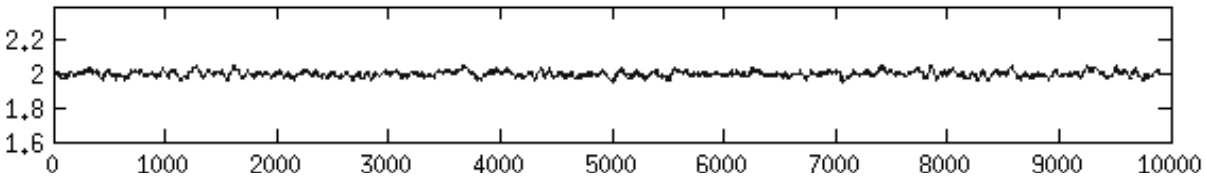



Figura 17 - Curva de valor filtrado a partir de sensor de valor-base 2 versus número da medição




Ao se comparar as curvas originais e as filtradas, pode-se ver claramente que o módulo de processamento e filtragem funcionam perfeitamente: para o valor-base de -1 o desvio padrão foi reduzido de 0,058 para 0,008 e para o valor 2 o desvio passou de 0,116 para 0,016, o que mostra o funcionamento do filtro de média. O fato de a média dos valores salvos ser igual à média da curva original mostra que, mesmo a 4 KHz, o fluxo do sistema funcionou corretamente.

# 6 Conclusão

Este trabalho apresentou uma biblioteca para desenvolvimento de sistemas microcontrolados de sensoriamento visando permitir uma maior variedade de tipos e uma maior velocidade e quantidade de sensores na rede. A biblioteca encapsula todo o núcleo de gerenciamento e comunicação inter-módulos, deixando a cargo do desenvolvedor apenas a implementação dos métodos para configuração e captura de dados dos sensores. A possibilidade de utilizar vários microcontroladores interconectados traz grande poder de processamento e de modularização ao sistema, permitindo-o cumprir os objetivos iniciais de alta velocidade, diversidade de sensores e facilidade de manutenção.

Foram realizados dois experimentos para validação da biblioteca e do seu desempenho. No primeiro teste pôde-se observar o funcionamento dos módulos de telemetria, sensoriamento, comando e controle, capturando dados do sensor conforme a configuração designada *on-the-fly* e os enviando via rádio para o computador, no qual puderam ser validados com sucesso. No teste de desempenho o sistema funcionou sem problemas a uma frequência duas vezes e meia maior que o mais rápido sistema equivalente apresentado (Emmendoerfer, 2008), lendo, filtrando e armazenando dados com médias pré-definidas de forma rápida e correta.

Os resultados positivos e o desenvolvimento modularizado da biblioteca a tornam uma excelente alternativa para o desenvolvimento de sistemas de navegação, contribuindo para a evolução e a expansão dos sistemas móveis autônomos.

# Referências bibliográficas


Analog Devices. **TMP3X Low Voltage Temperature Sensors**. EUA, 1996. 20 p.

Analog Devices. **ADXRS300 300 deg/s Single Chip Yaw Rate Gyro with Signal Conditioning**. EUA, 2004. 8 p.

Analog Devices. **ADXL345 3-Axis, 2/4/8/16 g Digital Accelerometer**. EUA, 2009. 40 p.

Bosch Sensortec. **BMA180 Digital, triaxial acceleration sensor**. EUA, 2009. 69 p.

Darby F. de A. Lopes; Marco Henrique Terra. **Estimativa de Atitude e Posição e Controle Robusto de um Helicóptero Autônomo**. 2010. 87 p. Dissertação de Mestrado, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos.

Dominique Paret. **The I2C Bus: From Theory to Practice**, 1 ed. EUA: John Wiley & Sons, 1997. ISBN 9780471962687

Douglas D. S. Santana; Celso M. Furukawa. **Navegação Terrestre Usando Unidade de Medição Inercial de Baixo Desempenho e Fusão Sensorial com Filtro de Kalman Adaptativo Suavizado**. 2011. 230 p. Tese de Doutorado, Escola Politécnica, Universidade de São Paulo, São Paulo.

Freescale Semiconductors. **MPXV7007 Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature and Calibrated**. EUA, 2011. 10 p.

Gustavo Emmendoerfer; Roger Gules. **Desenvolvimento e Implementação de um Sistema de Medição Inercial de Trajetórias de Dutos**. 2008. 159 p. Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná, Curitiba.

Honeywell Sensing and Control. **HIH5030/5031 Series Low Voltage Humidity Sensors**. EUA, 2010. 8 p.

José A. França Jr.; Jorge A. Morgado; Marcos F. D. Pinto. **Simulação e Implementação em Tempo Real de Sistemas de Navegação Inercial Integrados INS/GPS**. 2009. 113 p. Dissertação de Mestrado, Instituto Militar de Engenharia, Exército Brasileiro, Rio de Janeiro.

Luciano de Oliveira Neris; Onofre Trindade Jr.. **Um Piloto Automático para as Aeronaves do Projeto ARARA**. 2001. 113 p. Dissertação de Mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos.

Microchip Technology. **dsPIC33FJXXXMCX06/X08/X10 Data Sheet**. EUA, 2009. 340 p.

Microchip Technology. **16-bit Embedded Control Solutions**. EUA, 2011. 32 p.

Microchip Technology. **16-bit Embedded Control Solutions**. EUA, 2011. 32 p.

Motorola, Inc.. **SPI Block Guide**. EUA, 2003. 38 p.

Rodrigo Sauri Lavieri; André L. C. Fujarra. **Métodos de Navegação Inercial Aplicados a Lançamentos Submarinos**. 2011. 292 p. Dissertação de Mestrado, Escola Politécnica, Universidade de São Paulo, São Paulo.

Ronan Alves da Paixão; Paulo F. F. Rosa. **Aquisição e Processamento de Dados para Sistemas de Navegação Inercial**. 2010. 85 p. Trabalho de Conclusão de Curso, Instituto Militar de Engenharia, Exército Brasileiro, Rio de Janeiro.