Sistema de Acompanhamento de Missão para uma Plataforma Experimental em Robótica Subaquática

Marcos Vinicius Scholl[1], Carlos Rodrigues Rocha[2]

[1]Universidade Federal do Rio Grande (FURG) - Campus Carreiros: Av. Itália km 8 Bairro Carreiros, Rio Grande (RS) - Brasil

[2]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) - Campus Rio Grande, R. Eng. Alfredo Huch, 475 - 96201-460 - Rio Grande(RS) – Brasil

marcos.vinicius.scholl@gmail.com, carlos.rocha@riogrande.ifrs.edu.br

Abstract. The design and implementation aspects of a mission monitoring and control system for underwater unmanned vehicles are presented in this work. This system is part of a wider research project to develop an open experimental platform for underwater robotics, in order to stimulate more research studies in this area.

Resumo. Neste trabalho são apresentados aspectos de projeto e implementação de um sistema de acompanhamento de missão para veículos subaquáticos não tripulados. Este sistema é parte de um projeto de pesquisa que tem por objetivo desenvolver uma plataforma experimental aberta para robótica subaquática a fim de viabilizar um maior número de trabalhos nessa linha de pesquisa.

Introdução

O sistema de acompanhamento de missão é parte do projeto de uma plataforma experimental aberta para robótica subaquática. Esse projeto é desenvolvido por um grupo de trabalho formado por professores/pesquisadores do Campus Rio Grande do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS-Rio Grande), alunos do curso Técnico em Automação Industrial desse Instituto e alunos do curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Federal do Rio Grande (FURG).

O design da artigo seguiu o processo de desenvolvimento do projeto. Este será apresentado na próximas seções. A primeira seção demonstra o trabalho motivador deste projeto. A seção seguinte apresentará o modelo conceitual adotado para a plataforma. Seguindo, a descrição dos passos ao desenvolvimento do sistema em questão. Finalmente é apresentado o resultado das fases iniciais de desenvolvimento, mostrando uma visão do sistema proposto e então perspectivas de trabalho futuro será abordada em conclusão.

Trabalhos Relacionados

Uma revisão de trabalhos relacionados, tanto estrangeiros quanto brasileiros, está disponível em Rocha[13], trabalho motivador do desenvolvimento desse projeto, pela necessidade de validação experimental da metodologia de planejamento de movimento proposta.

1. Projeto Conceitual

Para estruturar o desenvolvimento do projeto, empregou-se a metodologia PRODIP (Processo de Desenvolvimento Integrado de Produtos) [BACK *et al.*, 2008]. A partir desta, uma análise inicial da plataforma resultou no levantamento de requisitos e posterior criação de concepções de soluções que atendessem aos requisitos identificados. Destas, a solução mais adequada aos requisitos principais (como ser uma plataforma aberta/livre e relativo baixo custo) foi selecionada para posterior desenvolvimento.

Nesta solução, a plataforma é formada por dois componentes, o veículo subaquático não tripulado (ou UUV, do inglês *Underwater Unmanned Vehicle*) e o sistema de acompanhamento de missão (ou MMCS, do inglês Mission Monitoring/Control System). A Figura 1 apresenta a visão geral da solução escolhida.

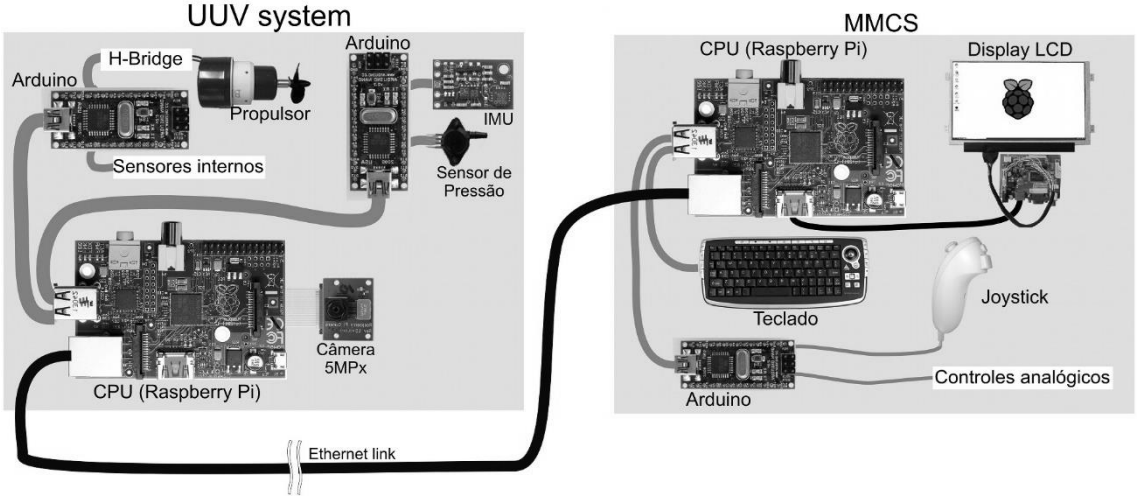

Figura 1 Visão geral do hardware da plataforma

Embora o conjunto seja formado pelo UUV e pelo MMCS, apenas este será abordado no restante do texto, por ser o foco do presente trabalho. Após a definição do hardware a ser empregado para a construção do MMCS, foi definida a plataforma de desenvolvimento do software de acompanhamento de missão e feito um projeto de sua estrutura, baseada no paradigma da orientação a objetos. Estes elementos serão detalhados a seguir.

2. Tecnologias de Hardware e Software Adotadas

Na construção do sistema de acompanhamento de missão, optou-se por projetar e construir um hardware portátil para poder ir a campo com facilidade. Diferentemente de um laptop/notebook, o hardware deve ser mais resistente às características de ambientes externos e ter controles analógicos. Em relação ao software, ele deveria ser constituído a partir de soluções compostas de Softwate Livre, sendo multiplataforma, capaz de rodar tanto no hardware construído para esse fim quanto em notebooks com sistemas operacionais conhecidos, caso se optasse por esse tipo de equipamento. Assim, foram escolhidas as tecnologias a seguir para a implementação do sistema de acompanhamento de missão.

2.1. Hardware

Arduino: Plataforma de prototipação muito popular, tem baixo custo, é aberto e tem grande base de conhecimento disponível. Assim, ele será adotado para a construção da interface analógica, com controles como um *joystick* wii Nunchuk e outros sensores. Além disso, haverá indicadores como leds e display LCD. O modelo utilizado na construção do módulo analógico é o Arduino Nano [ARDUINO, 2013].

Raspberry Pi: Esta placa de computação de baixo custo é do tipo *System-on-a-Chip* (SoC). Está sendo utilizado como a Unidade Central de Processamento do sistema de acompanhamento de missão, por ter baixo consumo de energia, capacidade de processamento suficiente para as necessidades do software, disponibilidade de rede Ethernet, e dimensões reduzidas. O RPi suporta diversos sistemas operacionais, entre eles distribuições Linux como o Raspbian, baseado no Debian, que permite tratá-lo como um computador similar aos PCs [RASPBERRY, 2013; THOMPSON; GREEN, 2014].

Entradas/Saídas: Para a interface com o usuário, foi adquirido um display LCD de 10 polegadas portátil de alta resolução com controlador de vídeo dedicado, leds indicadores de estado de operação do sistema, teclado com *trackball*, *joystick* com diversos eixos, além de alguns potenciômetros.

2.2. Software

O software da plataforma deveria ser extensível e facilmente adaptável às necessidades de experimentação. Assim, optou-se pela abordagem orientada a objetos para o projeto e posterior implementação. No caso do sistema de acompanhamento de missão, era necessário definir componentes visuais, ou *widgets*, para apresentarem informações de sensores e câmera do UUV. Além desses, deveriam haver classes que gerenciassem a comunicação entre o veículo e o acompanhamento de missão, necessária para a obtenção dos dados dos sensores e envio dos comandos para teleoperação do UUV. Assim, foi definido o diagrama de classes mostrado na Figura 2, que descreve as classes inicialmente definidas e o relacionamento entre elas.

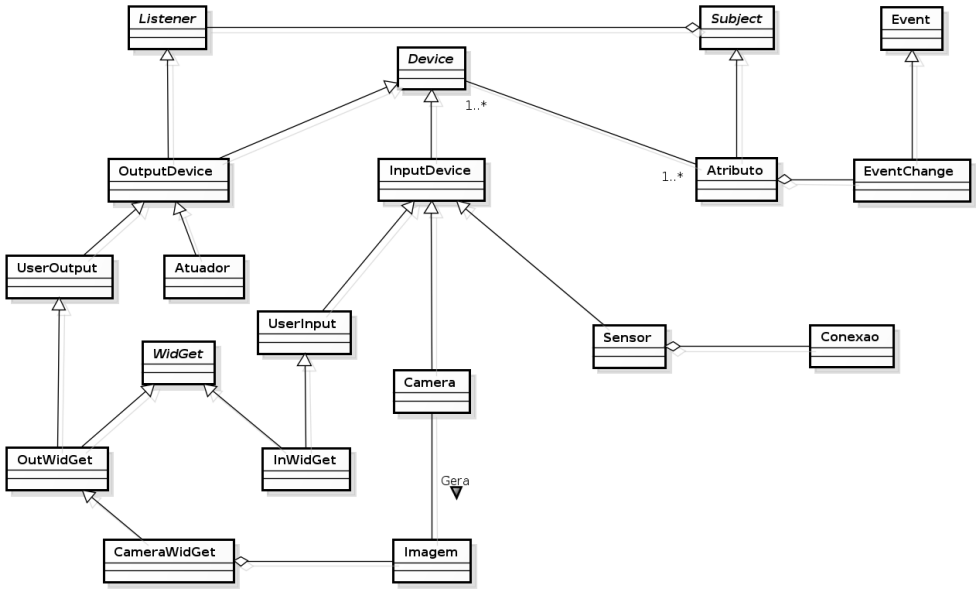

Figura 2 Diagrama de classes do sistema de acompanhamento de missão

A escolha da plataforma de desenvolvimento do software levou em consideração a necessidade dele ser multiplataforma e facilmente modificável/extensível, além de orientado a objetos. Além disso, a plataforma deveria ter disponibilidade de bibliotecas matemáticas e boa base de software/conhecimento existente. Assim, optou-se pela utilização de Python [PYTHON, 2014].

Além de ser uma linguagem multiplataforma (e disponível para o RPi, em particular), a linguagem é bastante utilizada no meio científico tanto para o desenvolvimento de sistemas quanto para integrar diferentes sistemas. A disponibilidade de bibliotecas numéricas/científicas como NumPy, SciPy, Matplotlib, NetworkX facilitam o desenvolvimento de códigos para experimentação nas linhas de pesquisa da robótica, como controle, planejamento de trajetórias, cinemática e dinâmica [OLIPHANT, 2007] [JONES *et al.*, 2007; HUNTER, 2007; HAGBERG; SCHULT; SWART, 2014]. A disponibilidade de uso da biblioteca OpenCV é relevante para o processamento de imagens da câmera do UUV [HOWSE, 2013]. Para a interface gráfica com o usuário, existem *binds* com diversas APIs, como GTK, WxWindows e Qt, sendo esta a adotada, pela utilização da biblioteca QWT, que além dos *widgets* usuais de interfaces gráficas ainda tem diversos componentes visuais customizáveis como escalas, *gauges* e *displays* [RATHMANN; WILGEN, 2014].

3. Desenvolvimento dos *Widgets*

Os componentes visuais, descendentes da classe *Widget*, foram definidos após um estudo de interfaces com usuários de veículos subaquáticos remotamente operados (ou ROV, do inglês *Remotely Operated Vehicle*). Deles, verificou-se o uso de *displays* textuais de dados numéricos, indicadores visuais binários (on/off, ou normal/problema), *gauges* ou escalas, bússola e horizonte artificial.

Alguns desses componentes já estavam disponíveis na biblioteca QWT, necessitando de algumas adaptações às necessidades do software. Outros componentes deveriam ser criados. Em ambos os casos, a tarefa foi bastante simplificada pelo uso do mecanismo de herança das classes já existentes. A principal modificação foi a atualização automática da apresentação dos dados, obtidos de provedores de dados (descendentes de InputDevice), que representariam os sensores do UUV junto ao sistema de acompanhamento de missão.

Os componentes implementados até o momento são brevemente apresentados a seguir. A Figura 3 mostra a interface homem-máquina (aqui denominada *cockpit*) criada com o uso desses componentes, para teste de suas funcionalidades.

3.1. Widget Gauge/Escala

É utilizado para mostrar dados numéricos entre limites mínimos e máximos. Na interface da Figura 3 (3.a), ele é utilizado para mostrar a profundidade, que é medida através de um sensor de pressão absoluta instalado no UUV. Assim, além da informação da profundidade em si, tem-se uma noção dos limites de operação do veículo.

3.2. Widget Bússola

A bússola apresenta a informação de direção do veículo (ou *heading*, em inglês). Ela é medida pelo magnetômetro de 3 eixos da unidade de medição inercial (ou IMU, do inglês

Inertial Measurement Unit) instalada no veículo. Essa informação é fundamental para a navegação do veículo. A bússola é mostrada na direita do *cockpit* (Figura 3.b).

3.3. Widget Horizonte Artificial

O horizonte artificial apresenta informações da atitude do veículo, correspondente à sua rolagem (ângulo do eixo transversal do UUV em relação ao plano horizontal) e à sua arfagem (ângulo do eixo longitudinal em relação ao plano horizontal) em uma simples representação. Ele é baseado nos instrumentos usados em aviação. Os dados correspondentes a essas variáveis são calculados a partir de medições feitas pela IMU usando seu acelerômetro e giroscópio de 3 eixos. Essas medições são integradas através de algoritmos de fusão sensorial, como os do software FreeIMU [VARESANO, 2014]. Na Figura 3, ele está no canto inferior direito (3.c).

3.4. Widget Informações

Esse é um componente binário, projetado para uso em indicadores de operação normal ou em exceção. Para tanto, uma instância de Informações pode definir um ícone relativo ao estado medido, que deve ter uma cor para operação normal (por *default*, verde) e outra para exceção/erro (vermelho, por *default*). Um agrupamento desses *widgets* é apresentado no canto inferior esquerdo do *cockpit* (Figura 3.d).

4. Interface Com o Usuário - *Cockpit*

O *cockpit* é a interface gráfica com o usuário do sistema de acompanhamento de missão. Nele são agrupados os *widgets*, que podem ser dispostos na tela de acordo com a necessidade do experimento, requisito desejado para o sistema (modificabilidade). Ele é na verdade uma instância da classe *MainWindow* da biblioteca Qt. Além dos componentes detalhados anteriormente, a visualização da câmera e os componentes de interação (botões, *sliders*) também comporão o *cockpit*. A Figura 3 mostra uma definição de *cockpit* utilizando os componentes até então implementados.

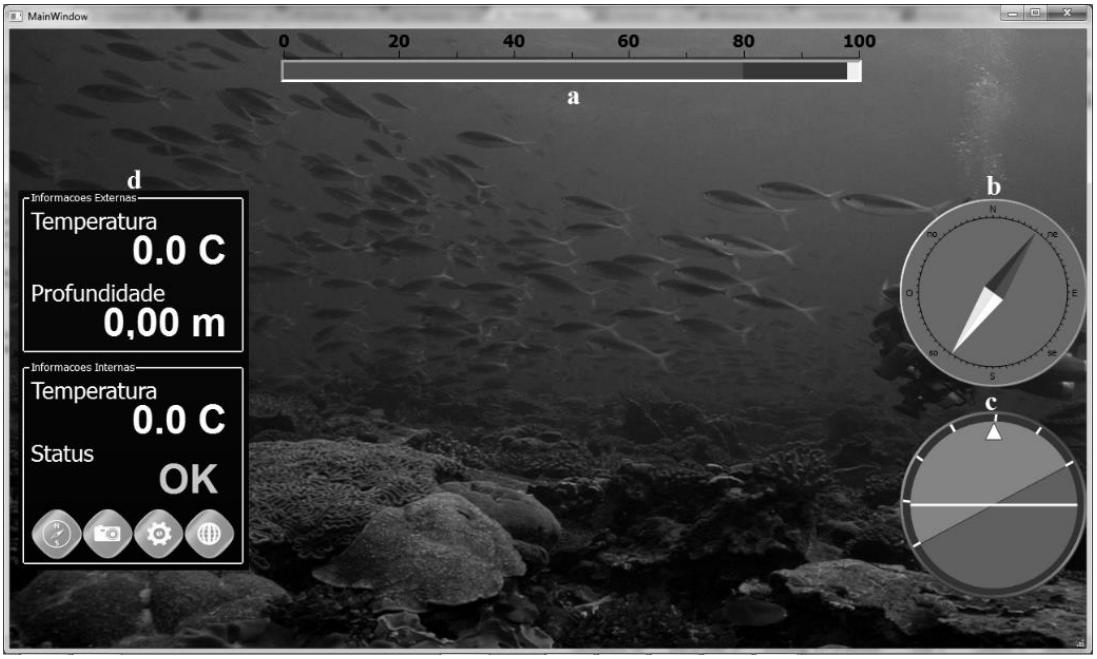

Figura 3 Cockpit do sistema de acompanhamento de missão

5. Apresentação e Uso do Vídeo do UUV

A câmera faz parte do subsistema do UUV. A imagem por ela captada é transmitida entre os RPi do UUV e da unidade de acompanhamento de missão por streaming de vídeo através da interface de rede Ethernet que compõe o cordão umbilical do veículo. Um *widget* de visualização do vídeo está sendo finalizado, junto com os testes da implementação do sistema de streaming, cuja dificuldade reside na obtenção de baixa latência associada à melhor qualidade de imagem possível.

6. Conclusão

Neste trabalho foram apresentados aspectos do projeto e implementação de um sistema de acompanhamento de missão para veículos subaquáticos não tripulados. Além das definições dos requisitos do sistema e conceituação adotada para atendê-los, foram discutidas as implementações feitas até o momento.

Este é um trabalho em andamento, com diferentes atividades sendo desenvolvidas em paralelo para a montagem do protótipo da plataforma experimental composta pelo veículo subaquático e pela unidade de acompanhamento de missão. Trabalhos futuros apresentarão a evolução do projeto e os resultados da experimentação com o protótipo, bem como derivações desse trabalho.

Por fim, observa-se que a plataforma experimental será disponibilizada como hardware/software livre, com o projeto acessível para quem quiser construir ou modificar a proposta inicial, constituindo assim uma contribuição à pesquisa experimental em robótica subaquática.

Referências

ARDUINO. **Arduino Nano**. (Online). Disponível em <http://arduino.cc/en/Main/arduinoBoardNano>. Acesso em 11/03/2014.

BACK et al. **Projeto Integrado de Produtos: Planejamento, Concepção e Modelagem**. São Paulo: Manole, 2008.

HAGBERG, A., SCHULT, D., SWART, P. **NetworkX: Software de alta produtividade para redes complexas**. (Online) Disponível em <http://networkx.github.io/>. Acesso em 11/03/2014.

HOWSE, J. **OpenCV Computer Vision with Python**. Packt Publishing, 2013.

HUNTER, J. D. Matplotlib: A 2D graphics environment. **Computing in Science and Engineering**, Vol. 9, No. 3. IEEE Computer Society. Maio 2007, pp 90-95. Disponível em <http://doi.ieeecomputersociety.org/10.1109/MCSE.2007.55>.

JONES, E. et al. **SCIPY: Open Source Scientific Tools for Python**. (Online) Disponível em <http://www.scipy.org/>. Acesso em 11/03/2014.

OLIPHANT, T. E. NumPy: Python for Scientific Computing. **Computing in Science and Engineering**, Vol. 9, No. 3. IEEE Computer Society. Maio 2007, pp 10-20. Disponível em <http://doi.ieeecomputersociety.org/10.1109/MCSE.2007.58>.

PYTHON Software Foundation. **Linguagem de programação Python**. (Online) Disponível em <http://www.python.org/>. Acesso em 11/03/2014.

RATHMANN, U. and WILGEN, J. (Online) Qwt User's Guide, **Qwt - Qt Widgets for Technical Applications**. Disponível em <http://qwt.sourceforge.net>. Acesso em 03/03/2014.

The RASPBERRY Pi Foundation. **Raspberry Pi**. (Online) Disponível em <http://www.raspberrypi.org/about>. Acesso em 15/12/2013.

THOMPSON, M., GREEN, P. **Raspbian**. (Online) Disponível em <http://www.raspbian.org/>. Acesso em: 11/03/2014.

VARESANO, F. **FreeIMU: an Open Hardware Framework for Orientation and Motion Sensing**. (Online) Disponível em <http://www.varesano.net/projects/hardware/FreeIMU>. Acesso em 07/02/2014.

[13] C. R. Rocha, "Motion planning of underwater intervention robotic systems based on screw theory - planejamento de movimento de sistemas robóticos de intervenção subaquática baseado na teoria de helicoides," PhD Thesis, Federal University of Santa Catarina, Florianópolis(SC), Brazil, 2012.