ARQUITETURA DE CONTROLE DE UM AUV BASEADA EM ENGENHARIA DE SISTEMAS

LUCIANO O. FREIRE[1], ETTORE A. DE BARROS[1].

1. *Laboratório de Veículos Não Tripulados, Departamento de Engenharia Mecatrônica, Escola Politécnica da Universidade de São Paulo*
Av. Prof. Professor Mello Moraes, 2231, sala MZ-08, 05508-030, São Paulo, SP, Brasil.
Fone(11)3091-6017
E-mails: luciano.ondir@gmail.com, eabarros@usp.br

Abstract — This work presents decentralized control architecture for an underwater vehicle developed with systems engineering concepts in order to assure parallel development, work continuity and positive synergy between the students. The first concept is the modularity allowing defining precise and smaller responsibility scopes and to identify and solve problems during development. The second is the technological plan, which aims at assuring the availability of the components for the life of the project and limits the diversity, easing the logistic aspects. The third aspect is interface management, which allows each stakeholder to work independently inside his perimeter, respecting the interfaces which must be as simple and precise as possible. The fourth aspect is functional classification to identify functions that need to be simple and high quality to assure the safety of the plant while the great majority of the function can be developed with normal quality. This architecture aims to be low cost, extensible and versatile. For low cost we mean cheap and based on off the shelf components, free and largely known software tools, like C language, GNU compiler, etc. which requires low training time for an average undergraduate or graduate engineer student, which means low investment and time for adding a new functionality to the system. An implementation of an Autonomous Underwater Vehicle (AUV) control system is done and some preliminary experimental results are shown.

Keywords— Autonomous robots, Embedded intelligent systems, System engineering.

Resumo— Este trabalho apresenta uma arquitetura de controle descentralizada para um AUV desenvolvida com conceitos da engenharia de sistemas para assegurar desenvolvimento paralelo, continuidade do trabalho e sinergia positiva entre os alunos. O primeiro conceito é a modularidade para definir escopos de responsabilidade precisos e pequenos, permitindo que os problemas possam ser identificados e resolvidos durante o desenvolvimento. O segundo é o plano tecnológico, que visa assegurar a disponibilidade dos componentes ao longo do projeto e reduz a diversidade tecnológica, facilitando os aspectos logísticos. O terceiro aspecto é a definição de interfaces, permitindo o trabalho independente dentro de cada perímetro, respeitando as interfaces, que devem ser o mais simples e precisas possível. O quarto aspecto é a classificação funcional para identificar funcionalidades que precisam ser simples e de alta qualidade para proteger a planta enquanto a grande maioria das funcionalidades pode ser desenvolvida com qualidade normal. Essa arquitetura tem os objetivos de ter baixo custo, de ser extensível e versátil. Baixo custo significa componentes baratos de prateleira, ferramentas gratuitas e amplamente conhecidas de desenvolvimento de software, como linguagem C e compilador GNU, que requerem pouco tempo de treinamento para um estudante de engenharia, o que significa pouco esforço para adicionar novas funcionalidades. Um sistema de controle para veículos submarinos autônomos (AUV) foi feito e resultados experimentais preliminares são mostrados.

Palavras-chave— Robótica autônoma, Sistemas inteligentes embarcados, engenharia de sistemas.

1 Introdução

Os AUVs (Autonomous Underwater Vehicles) têm conquistado um mercado cada vez maior nas áreas de exploração econômica, pesquisa científica e missões militares, nos cenários oceânicos, devido à sua flexibilidade e baixo custo de suas operações se comparado às soluções mais tradicionais. Devido à sua operação não assistida, estes veículos incluem um sistema de controle embarcado relativamente sofisticado. Por outro lado, devido à natureza interdisciplinar do projeto de AUVs e sua crescente complexidade, seu desenvolvimento é um campo fértil para a aplicação de ideias oriundas da engenharia de sistemas. Mesmo no caso de pequenos veículos desenvolvidos por universidades, existem problemas de engenharia mecânica (vasos de pressão e selos), engenharia naval (arranjo, hidrodinâmica e integração casco – motor – hélice), engenharia elétrica (atuadores, distribuição e armazenamento de energia), engenharia da computação (processadores e redes de dados), engenharia de controle (algoritmos de controle e arquitetura de controle) e engenharia de *software* (*drivers* de baixo nível e funções de alto nível) para serem tratados. Esse número de competências e assuntos resulta em uma complexidade muito grande, que, comumente impede que uma única pessoa consiga executar um projeto satisfatório de um AUV.

Isso significa que um projeto bem consolidado possui muito desenvolvimento acumulado, feito por diferentes pessoas. Tal projeto não pode ser feito no âmbito de um trabalho acadêmico sem prejuízo da finalidade do mesmo, ainda que o desenvolvimento do AUV seja necessário para a condução da pesquisa. Entretanto, a própria interdisciplinaridade dos AUVs pode jogar a favor da pesquisa científica, pois uma única plataforma pode servir de base para a produção de conhecimento em vários campos do saber humano. Mas, para isso, o reuso de soluções é fundamental para reduzir custos e tempo. Reuso, por sua vez, requer modularidade e gerenciamento de

interfaces, o que permite que muitos contribuidores, sejam do meio acadêmico ou não, possam trabalhar em paralelo. Outro requisito fundamental é a flexibilidade, que é ter uma solução técnica que possa facilmente ser modificada para absorver novas funcionalidades. É importante reduzir ao máximo o tempo de aprendizado e treinamento dos programadores no seu processo de integração à equipe de desenvolvimento e utilização da arquitetura de controle, tentando-se tirar proveito da formação que ele já traz de suas atividades acadêmicas. A experiência demonstra que empregar ferramentas amplamente conhecidas, apesar de não serem as mais produtivas em ambientes empresariais, resulta em melhor rendimento, haja vista que os alunos podem começar a trabalhar depois de muito pouco tempo de treinamento, e, tomados cuidados para assegurar a retenção do conhecimento, o trabalho vai sendo integrado ao capital intelectual do laboratório.

Existem na literatura dois trabalhos bastante interessantes nesse sentido. No primeiro, Chitre (2008) apresenta a arquitetura *backseat* que separa as funcionalidades intrínsecas do veículo em um computador chamado *frontseat* e as funcionalidades da missão em outro computador (*backseat*). No segundo Eickstedt (2010) apresenta a DSAAV (*Distributed Software Architecture for Autonomous Vehicules*), que emprega *hardware* e *software* modulares e facilmente reconfiguráveis.

Entretanto, não existe na literatura aberta a aplicação sistemática de conceitos da engenharia de sistemas para desenvolvimento de veículos autônomos para aumentar a eficiência na produção do conhecimento, o que vem a ser a proposta deste trabalho.

2 Propósito

O objetivo deste trabalho é propor e analisar, sob o ponto de vista da engenharia de sistemas, os princípios norteadores da condução de projetos no âmbito de um laboratório universitário com a finalidade de permitir o trabalho em paralelo de grupos multidisciplinares e o acúmulo de conhecimento, e a partir daí, estabelecer os requisitos arquiteturais, tecnológicos e ferramentas de desenvolvimento para a arquitetura de controle de um AUV de baixo nível com alta confiabilidade e segurança. Esta arquitetura servirá como base para a implementação de funções de alto nível usando ferramentas tais como o ROS, proposto por Quigley, M. et al (2009).

3 Métodos

Engenharia de sistemas é um campo interdisciplinar da engenharia com foco na maneira de gerenciar e projetar sistemas complexos ao longo de todo o seu ciclo de vida. Questões como logística, coordenação de diferentes equipes e automação se tornam exponencialmente mais difíceis com o aumento da complexidade. A engenharia de sistemas lida então com métodos de trabalho e ferramentas para gerenciar riscos e envolve disciplinas tanto técnicas quanto centradas no ser humano, como engenharia de controle, engenharia industrial, estudo de organizações e gerenciamento de projeto. A NASA (2007) publica um manual de engenharia de sistemas que foi basilar para este trabalho. Evidentemente, foram escolhidos apenas alguns conceitos, julgados aplicáveis aos AUVs.

3.1 Modularidade

Para dominar a complexidade e dividir as tarefas para cada pessoa ou equipe, o sistema global precisa ser dividido em partes menores, denominadas subsistemas, permitindo desenvolvimento paralelo, e o agrupamento de famílias de competências similares dentro de cada subsistema.

O grande problema da modularidade é a questão da posterior integração, pois, sempre que houver uma interface entre dois subsistemas desenvolvidos por pessoas diferentes, haverá necessidade de uma atividade de integração dos dois subsistemas. O motivo é a dificuldade de comunicação entre diversas equipes, cada uma com uma área de conhecimento diferente. Por isso, a divisão de responsabilidades deve estar bem clara para evitar conflitos.

Assim sendo, a definição correta das interfaces é algo crítico para um projeto, para que cada pessoa possa desenvolver seu sistema de acordo com as especificações globais e as restrições impostas pelas interfaces.

A outra decorrência da modularidade é a evidente necessidade da redução de interfaces, seja em número, seja em complexidade, pois cada interface implica em trabalho adicional de coordenação, atividades de integração, riscos para o projeto, aumento de tempo de desenvolvimento e de custos.

O custo de desenvolvimento é elevado. Assim sendo, sempre que possível é interessante fazer subsistema genéricos, de modo que eles possam ser reaplicados a diversos projetos, o que é denominado projeto para reuso.

3.2 Plano tecnológico

Atualmente, existe uma evolução tecnológica muito veloz e existem muitas opções disponíveis no mercado. Ao se adotar uma estrutura hierárquica de projeto, onde os desenvolvedores de subsistemas tem liberdade para encontrar a solução que lhes parece mais adequada, existe a tendência natural de emprego de muitos tipos de tecnologias diferentes para um mesmo tipo de problema, como por exemplo, conectores e cabos elétricos.

Essa tendência cria problemas logísticos, que é manter um estoque de uma grande variedade de

componentes diferentes que, no fim, fazem a mesma coisa e provavelmente não são intercambiáveis.

Outro fator é a redução do número de tecnologias, pois havendo menor quantidade de componentes diferentes, menos tempo de treinamento é requerido de cada participante. Cada tecnologia tem um modo de funcionamento que às vezes requer um tempo considerável de aprendizado. Se dois desenvolvedores usam tecnologias diferentes, por exemplo, linguagem de programação, ambiente de desenvolvimento, microprocessadores, sensores, entre outros, fica muito difícil realocar pessoal de um subsistema para outro, pois será necessário muito tempo de aprendizado.

Existe ainda o problema da obsolescência. Algumas vezes uma tecnologia tem um desempenho muito interessante para uma dada aplicação, porém se ela deixar de ser fabricada, o projeto encontrará contratempos para sanar o problema.

Se for definido um pequeno conjunto de soluções admissíveis para o projeto logo de início, todos esses problemas podem ser minimizados, reduzindo o custo devido à economia de escala e redução de estoques e os atrasos no projeto devido à diminuição de variáveis imprevisíveis.

Entretanto, não basta que o número de tipos de componentes seja reduzido para que o andamento do projeto esteja assegurado. Os seguintes aspectos devem ser levados em consideração na escolha das tecnologias empregadas:

a. Maturidade: é interessante buscar tecnologias maduras e amplamente difundidas, mas que ainda não começaram a cair em desuso. Essa abordagem reduz o risco de ter que tratar problemas de obsolescência. Exemplos de tecnologias maduras são o padrão Ethernet e a rede CAN.

b. Conhecimento da tecnologia: bastante relacionado à maturidade, é a facilidade com que pode-se obter pessoal com capacidade de lidar com a tecnologia em questão, seja por meio de admissão ou treinamento do pessoal.

c. Desempenho técnico: Deve-se combater a tendência de buscar um ponto de desempenho sacrificando todos os outros fatores e lembrar que, após atingir o nível mínimo de desempenho requerido pela aplicação, o excesso de capacidade pode ser inútil.

d. Custo: Ao escolher uma tecnologia é muito importante atentar para o modo de emprego da mesma. Se o ambiente de uso for hostil por diversas razões, tais como manuseio, atmosfera salina, choques, pessoal inexperiente, o emprego de tecnologias dispendiosas poderá acarretar grandes gastos imprevistos.

e. Disponibilidade no mercado local e tempo de aquisição: um fator que acrescenta muita confiança ao projeto é ter a disponibilidade, no mercado local, dos componentes empregados. Isso permite reduzir estoques e evita atrasos devidos a falha de componentes, que podem ser rapidamente repostos.

3.3 Gerenciamento de interfaces

Uma vez que o projeto progride definindo subsistemas e as interfaces entre os subsistemas, deve-se atentar para o fato de que cada subsistema diz respeito a apenas um responsável e cada interface diz respeito a, no mínimo, dois responsáveis. Assim sendo, modificar interfaces é pelo menos duas vezes mais oneroso do que modificar um subsistema. À parte a solução trivial de reduzir o número de interfaces, a solução clássica é manter um controle rígido sobre o respeito às interfaces e à clarificação da natureza das mesmas.

Deve-se conceituar como interface tudo o que afeta direta ou indiretamente outros sistemas. Em AUVs, uma vez que existe o problema de equilíbrio de peso e empuxo, o peso dos subsistemas se torna uma interface crítica. Existem muitos tipos de interfaces, tais como:

- Peso (massa e posição do centro de massa);
- Volume (volume total e dimensões);
- Fixação mecânica (calços, jazentes, apoios, prendedores);
- Alimentação elétrica (tipo, tensão, tipo de cabeamento, conectores);
- Dados (protocolo, taxa de transmissão, meio físico);
- Fluidos (flanges, tubulações, tipo de fluido, pressão, temperatura);
- Transferência de calor (dissipação térmica, refrigeração);
- Penetrações estruturais (furos, prensa cabos, janelas de inspeção;
- Interface entre partes de software.

3.4 Classificação funcional

Entende-se que, mesmo o AUV sendo muito mais barato do que os competidores, como navios de superfície, a perda de um AUV implica em custos elevados e atrasos em um projeto, o que não é desejável.

Todo tipo de equipamento tem uma taxa de falha associada. Essa taxa de falha geralmente é mais elevada no início da vida útil do equipamento devido a problemas de fabricação. No final da vida útil, essa taxa também tem a tendência de se elevar devido ao desgaste.

No projeto de sistemas que afetam a segurança e que devem ter uma confiabilidade elevada, cuidados especiais com a fabricação e projeto devem ser tomados. Tais cuidados têm um custo elevado e que é proporcional ao nível de segurança. Entretanto, reduzir a taxa de falhas devido ao desgaste pode ser praticamente impossível ou muito caro.

Se empregarmos uma arquitetura na qual a função de prover segurança é completamente separada da função normal do equipamento, tal como previsto pela IEC (2010), será possível empregar cuidados especiais de projeto e fabricação somente em uma pequena parte do equipamento, reduzindo o custo e mantendo a segurança. Além disso, uma vez que o equipamento que provê a função de segurança não trabalha o tempo todo, são reduzidos os problemas de desgaste.

Adicionalmente, funções de segurança requerem equipamentos simples, são facilmente identificáveis e não variam muito de geração para geração do sistema. Assim, funções de segurança podem ser reusadas com maior frequência do que as funções normais de controle, que variam bastante de versão para versão. Através do reuso, pode-se reduzir o custo das funções de segurança. Assim ocorre uma economia de esforço muito grande na atualização do sistema de controle normal, que não precisa de um controle de qualidade.

4 Resultados

Segue uma descrição sucinta da arquitetura de controle desenvolvida em função de cada aspecto.

4.1 Modularidade

A base da arquitetura é a ideia proposta por Zhang (2008), que é empregar uma rede CAN ligando vários nós que gerenciam sensores e atuadores e um nó de processamento central capaz de fazer cálculos mais complexos, tais como navegação, mapeamento e planejamento de trajetória, conforme representado na Figura 1.

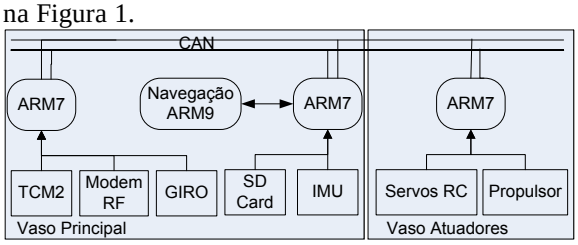

Figura 1 Arquitetura baseada em rede CAN

Cada nó seria desenvolvido por uma pessoa diferente e estaria relacionado a uma funcionalidade do AUV (exemplos: propulsão, governo, sensores inerciais, navegação). Assim cada aluno tem um escopo limitado de contribuição para o AUV e cada contribuição pode ser tratada como uma caixa preta.

A ideia central da arquitetura de software é prover uma biblioteca de funções para facilitar posteriores desenvolvimentos de objetos de alto nível. Para organizar a arquitetura, foram definidas camadas, nas quais os objetos podem usar funções dos objetos da própria camada ou das camadas adjacentes. Uma apresentação sumária dos principais objetos e camadas pode ser vista na Figura 2.

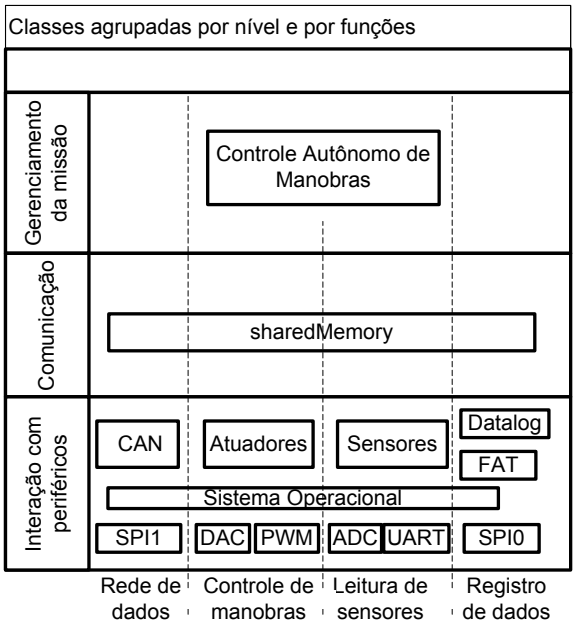

Figura 2 Arquitetura de *software*

Essas funções básicas foram desenvolvidas buscando um padrão em consonância com o padrão da MISRA (1994) para linguagem C para prover robustez e confiabilidade. Assim foi construído uma arquitetura modular tanto no hardware como no software.

4.2 Plano tecnológico

Microprocessadores possuem hoje um extenso leque de opções para sua programação, que inclui comumente ferramentas gratuitas. A linguagem mais comum é o ANSI C, existindo até sistemas operacionais codificados em C. Além disso, C faz parte do currículo em muitas universidades. Por isso, foi adotada a linguagem C e o ECLIPSE com o *plugin* CDT como ambiente de desenvolvimento, e o GNU-GCC como compilador. Tais ferramentas foram escolhidas por serem gratuitas, abertas, de fácil uso e conhecidas.

Decidiu-se empregar um sistema operacional para facilitar a modularidade e a orientação a objetos dentro de cada módulo, permitindo que o desenvolvedor possa dividir o trabalho em várias tarefas independentes, evitando problemas de interação. O sistema operacional adotado, seguindo Zhang (2008), foi o µC-OSII, criado por Labrosse (2002), levando-se em consideração o fato de ser aberto e gratuito para universidades, além de ser embarcável em microprocessadores. O µC-OSII tem uma estrutura bastante simples e é bastante didático. Vale a pena ressaltar que é um sistema operacional preemptivo de tempo real, certificado para equipamentos médicos e aeronáuticos.

O processador escolhido o LPC2148 da NXP *semiconductors*, que possui arquitetura ARM7 e que apresenta melhor relação custo-benefício em termos de memória permanente, memória volátil e número de periféricos. Oferece ainda a disponibilidade ime-

diata de módulos produzidos no País a um baixo custo. Uma vez que esse microprocessador possuía recursos de *hardware* que possibilitava a interface com todos os equipamentos disponíveis, seria necessário apenas um modelo de *hardware* para todas as tarefas. Além disso, o poder computacional desse *hardware* permite a aplicação de filtros mais elementares e tarefas mais complexas.

A arquitetura de *hardware* foi baseada em um pequeno número de tipos de placas que podem ser empilhadas, com um barramento paralelo vertical, de maneira similar ao padrão PC-104 e de maneira que cada nó possa ter todas as funcionalidades necessárias.

Foi decidido adotar uma rede CAN porque considera-se que a rede CAN é bem mais simples, necessitando apenas de um barramento de dois fios, enquanto que a rede *ethernet* requer um *hub switch* ao qual todos os nós são ligados, produzindo fisicamente uma rede na forma de estrela, além de exigir um cabo com mais fios.

O emprego da rede CAN se justifica pelo reduzido cabeamento requerido, pequeno espaço disponível a bordo, característica de tempo real, amplo uso na indústria automobilística, presença no mercado local, maturidade da tecnologia (evitando problemas de obsolescência), baixo custo e simplicidade e por ser satisfatória em termos de desempenho para comunicação com sensores e atuadores.

4.3 Gerenciamento de interfaces

Cada nó tem uma única interface de dados com o resto do sistema (o barramento CAN). Além disso, existe a alimentação de 5Vdc e os cabos conectando cada nó aos respectivos sensores e atuadores. Todos os cabos passam por conectores polarizados e coloridos do tipo *Mike* para evitar danos por conexão errada, como visto na Figura 3.

Figura 3 Caixa protetora dos circuitos eletrônicos

Cada nó fica envolto por uma caixa de plástico padrão para evitar que a atmosfera salina do mar atinja os circuitos eletrônicos, conferindo maior robustez. Cada aluno fica responsável por definir a interface com os sensores e atuadores, mas interface de alimentação e barramento CAN não pode ser modificada.

Internamente ao nó, existem várias placas do tamanho de meio cartão de crédito empilhadas usando um barramento paralelo de 64 pinos. Caso o aluno veja a necessidade de desenvolver algum novo tipo de placa, ele deverá obedecer ao padrão de tamanho e barramento de 64 pinos.

4.4 Classificação funcional

Decidiu-se estabelecer dois níveis de segurança. O primeiro, denominado crítico, é ligado ao barramento CAN e envolve os nós de processamento que potencialmente afetam a sobrevivência do AUV, que serão desenvolvidos com um padrão de qualidade mais elevado e empregar um hardware robusto.

O segundo nível, denominado normal, envolve outros sistemas que serão desenvolvidos sem imposições de hardware ou padrões de programação, tais como computadores do tipo ARM9 que farão a navegação usando filtros de Kalman. Tais sistemas não serão ligados diretamente ao barramento CAN, mas usarão uma interface de dados para se comunicar com um nó do nível crítico, como visto na Figura 1. Essa interface de dados deve ser desenvolvida de modo a assegurar que uma falha no nível normal não se propague para o nível crítico.

Para atingir a qualidade de software nos equipamentos críticos, foi adotada a MISRA (1994), pois esta norma provê uma série de recomendações que tornam o código mais legível, evitam os problemas mais conhecidos da linguagem e tornam mais raros os erros de falha humana.

4 Discussões

A funcionalidade do AUV foi verificada durante os testes em uma piscina. A estação de base na superfície podia iniciar e interromper manobras pré-programadas através de comandos transmitidos através de um link de rádio com um nó no vaso principal. Os dados obtidos de leme e rumo de uma das manobras pode ser visto na Figura 4.

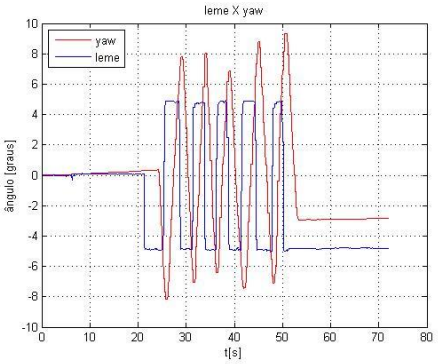

Figura 4 Comparação do ângulo de leme e do yaw

Tendo em conta o comprimento limitado da piscina, as manobras na piscina foram realizados a uma velocidade de 1 m / s. Um controlador de profundidade foi implementado usando um algoritmo de alocação de pólos. A profundidade foi mantida em torno de um valor constante de 0,8 m durante as manobras em ziguezague em relação ao

plano horizontal. A arquitetura apresentada até agora foi entregue em 2010.

A principal validação desse trabalho foi a continuidade do desenvolvimento por outros alunos do Laboratório de Veículos não Tripulados. A versão de 2012 da arquitetura de controle apresenta uma série de novas funcionalidades, deixando o veículo bem mais adaptado à sua função de executar ensaios livres. Entre as novas funcionalidades estão:

- Comando de início de manobra por meio de uma torção no leme;
- Eliminação do cordão umbilical, havendo troca de informações com a estação base somente na superfície;
- Estação base apenas configura o tipo de manobra a ser executada e descarrega os dados obtidos no ensaio (o veículo opera de maneira realmente autônoma agora);
- Novos tipos de controle de profundidade e de rumo foram criados;
- Novas manobras foram adicionadas;
- Novos sensores foram incorporados;

A manobra de zigzag foi aperfeiçoada, incorporando maior número de passos, como pode ser visto na Figura 5 e na Figura 6.

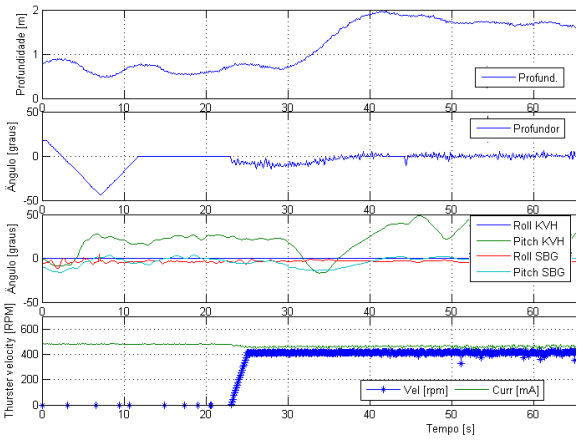

Figura 5 Manobra de zigzag. Variáveis do plano vertical.

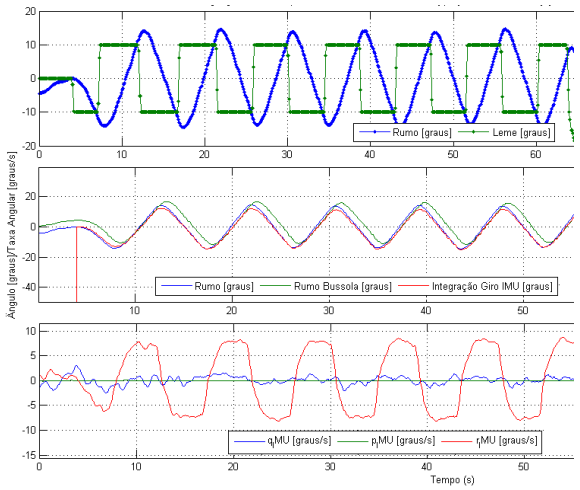

Figura 6 Manobra de zigzag. Variáveis do plano horizontal.

5 Conclusões

A atividade de pesquisa é feita com pessoas, logo o projeto dos meios de apoio deve ser centrado no elemento humano, determinando arquiteturas e influindo na escolha de tecnologias para que haja sinergia positiva entre os membros do grupo.

Para que seja desenvolvido continuamente um veículo autônomo submarino dentro de um laboratório universitário alguns conceitos da engenharia de sistemas precisam ser levados em conta.

Foi desenvolvida uma arquitetura dentro de alguns preceitos da engenharia de sistemas com o intuito de assegurar a continuidade do trabalho. A consecução de tal objetivo foi demonstrada por meio da efetiva realização de novas versões por outros alunos.

Como trabalhos futuros prevê-se a inclusão de nós para realização de navegação e mapeamento usando o ROS, proposto por Quigley, M. et al (2009).

Referências Bibliográficas

CHITRE, M. (2008). DSAAV - A distributed software architecture for autonomous vehicles. OCEANS 2008, pp 1 – 10. DOI: 10.1109/OCEANS.2008.5151848

EICKSTEDT, D. P., SIDELEAU, S. R. (2010) The Backseat control architecture for autonomous robotic vehicles: A case study with the Iver2 AUV. Marine Technology Society Journal, Vol. 44, No. 4, pp 42 -54. DOI: 10.4031/MTSJ.44.4.1

IEC (2010). Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems IEC61508, IEC, 2010.

LABROSSE, J. J. (2002), "µC/OS-II, The Real-Time Kernel", CMP Books, Kansas - USA, second edition, 2002.

MISRA (1994). Developement Guidelines for Vehicle Based Software, Motor Industry Software Reliability Association, 1994.

NASA (2007). NASA Systems Engineering Handbook, Revision 1, NASA/SP-2007-6105, NASA.

QUIGLEY, M. et al (2009). ROS: an open-source Robot Operating System, ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

ZHANG, L. et al. (2008). Design and experiment of automatic pilot for long range AUVs. 2008 3rd IEEE Conference on Industrial Electronics and Applications, pp 1824-1827.

