PROPOSTA DE APLICAÇÕES INTEGRANDO O SISTEMA OPERACIONAL CONTIKI COM A PLATAFORMA DE TESTES UNITÁRIOS CUNIT

Higor Alexandre de Castro, Márcio José da Cunha, Renato Santos Carrijo
Universidade Federal de Uberlândia – UFU, Faculdade de Engenharia Elétrica – FEELT, Uberlândia – Minas Gerais
higor0011@gmail.com, renato@eletrica.ufu.br, mjcunha@eletrica.ufu.br

***Resumo* – Em muitos processos industriais atualmente faz-se necessários sistemas que trabalham em tempo determinístico, e uma solução é a utilização de sistemas operacionais que podem gerenciar as atividades de equipamentos e de sensoriamento. Além disso, por motivos de infraestrutura, cada vez é mais comum o uso de redes sem fio no sensoriamento fabril. O objetivo deste trabalho é apresentar uma nova proposta de integração do sistema operacional de tempo real Contiki juntamente com a plataforma de testes unitários CUnit.**

O Contiki é um sistema operacional de código aberto para Internet das Coisas – *Internet of Things* (IoT), destinado principalmente a plataformas de prototipagem sem fio de baixa potencia e que juntamente com a ferramenta de teste unitário CUnit, será possível o desenvolvimento de equipamentos de baixo custo e robustez provido pelos testes unitários.

***Palavras-Chave* – Contiki, CUnit, Internet das Coisas, Rede de sensores sem fio, RTOS, Teste Unitário.**

APPLICATIONS PROPOSAL INTEGRATING THE OPERATIONAL SYSTEM CONTIKI WITH THE UNIT TEST PLATAFORM CUNIT

***Abstract* - In many industrial processes currently makes up necessary systems working in deterministic time, and a solution is use operating systems that can manage the activities of equipment and sensing. In addition, for infrastructure reasons, it is increasingly common to use wireless networks in industrial sensing. The aim of this paper is to present a new proposal for the operating system integrating real time Contiki along with unit testing platform Cunit.**

The Contiki is an open source operating system Internet of Things (IoT), primarily developed for low power wireless prototyping platforms and along with the unit test tool Cunit, it will be possible the development of low cost equipment and robustness provided by unit tests.

***Keywords* – Contiki, CUnit, Internet of Things, RTOS, Unit Test, Wireless Sensor Network.**

XIV CEEL - ISSN 2178-8308
03 a 07 de Outubro de 2016
Universidade Federal de Uberlândia - UFU
Uberlândia - Minas Gerais - Brasil

I. INTRODUÇÃO

Com os avanços tecnológicos na área de controladores, sensores e circuitos integrados, foi possível a criação das redes de sensores sem fio principalmente no que diz respeito a funções de monitoramento.

Tomando como exemplo o chão de fabrica, onde várias maquinas se comunicam em um processo industrial, e que devido a limitações físicas se torna difícil estabelecer o meio físico de comunicação entre elas, redes de sensores sem fio são capazes de sanar o problema.

Alguns processos industriais devem responder a estímulos internos e externos num tempo suficientemente curto e compatível com a urgência e importância do evento, Além disso, uma rede de sensores sem fio geralmente trabalham de forma autônoma e requer cooperação para executar as tarefas definidas naquela rede. Devido a mudança do tipo de comunicação, os algoritmos e protocolos tradicionais de devem ser revistos antes de serem aplicados.

A fim de testar a confiabilidade de comunicação e dos dados, uma ferramenta de Testes Unitários pode ser aplicada no desenvolvimento de uma rede de sensores sem fio (RSSF), ou WSN (*Wireless Sensor Network*), validando assim de maneira consistente qualquer projeto.

Uma ferramenta de teste unitário consiste em automatizar um teste cuja finalidade é executar testes apenas em uma pequena unidade isolada do sistema geralmente métodos e funções.

Com a ferramenta de teste unitário ou chamado também de teste de unidade é possível testar exaustivamente funções de maneira rápida e eficiente e que garantem uma saída válida. A ferramenta gera também, um relatório da falha, e onde que a mesma ocorreu.

Juntamente com o cenário de testes unitários, rede de sensores sem fio e sistemas de tempo real, surge a proposta de conciliação entre as plataformas e que serão mostradas a diante neste artigo.

II. SISTEMAS DE TEMPO REAL

Para processos determinísticos, ou seja, devem apresentar uma resposta em um determinado espaço de tempo, faz-se necessário o uso de sistemas operacionais de tempo real. Como por exemplo, o *airbag* automotivo, em que o tempo de acionamento é determinante para que faça corretamente seu papel de segurança [1].

Sistemas operacionais por definição são responsáveis por gerenciar todos os recursos de hardware e aplicações. Em geral os principais recursos a serem gerenciados são o processador, a memória e os dispositivos de entrada e saída.

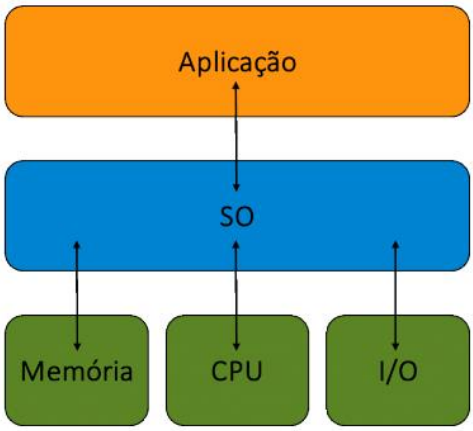


Fig. 1. Gerenciamento de hardware e software[2]

No caso de um sistema operacional de tempo real (RTOS – *Real Time Operational System*) é projetado especialmente para rodar aplicações que exigem extrema precisão e alto grau de confiabilidade. O RTOS é capaz de priorizar tarefas, sendo as mais criticas possuem maior controle do processador quando necessário.

No ambiente embarcado, existem diversas restrições que impedem o uso de sistemas operacionais convencionais. Situações como quantidade de memória, capacidade de processamento, ou ainda a plataforma hardware utilizada não possui a implementação do sistema operacional desejado. Esses impedimentos são mais comuns em sistemas de médio e baixo custo.

Visando atender esses dispositivos foram desenvolvidos diversos sistemas operacionais de tempo real, um deles, e que será abordado neste artigo é o RTOS Contiki.

III. CONTIKI

Contiki[3] é um sistema operacional de código aberto designado a Internet das Coisas (*IOT – Internet of Things*) e com enfoque maior em sistemas de baixo consumo e pouca memória. Por conta disso, ele consome apenas aproximadamente 10 Kbytes de memória RAM e 30 Kbytes de memória ROM.

O sistema operacional Contiki fornece também mecanismos de rede, com os protocolos UDP, TCP e HTTP. Suporta totalmente o padrão IPv6 e IPv4, além das novas normas de baixa potência como 6LowPAN, RPL e Coap.

Na plataforma Contiki é possível simular redes através da interface visual Cooja, que permite aos desenvolvedores ver em detalhes os aplicativos executados e dispositivos de hardware totalmente emulados, como é mostrado na figura a seguir:

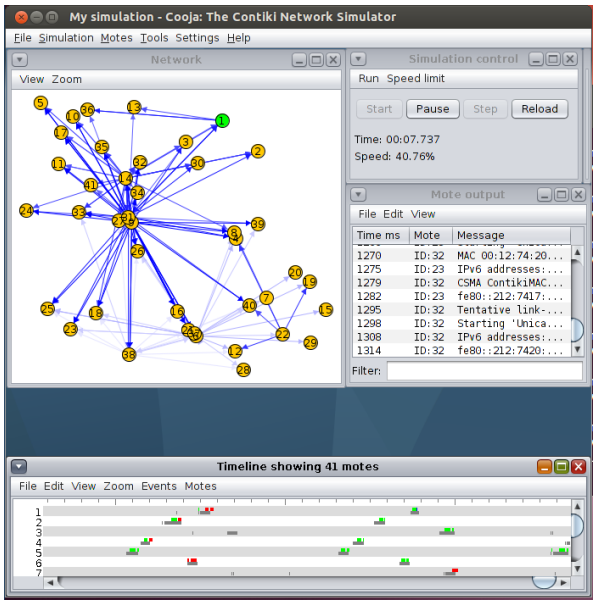


Fig. 2. Simulação de uma rede no software Cooja[4].

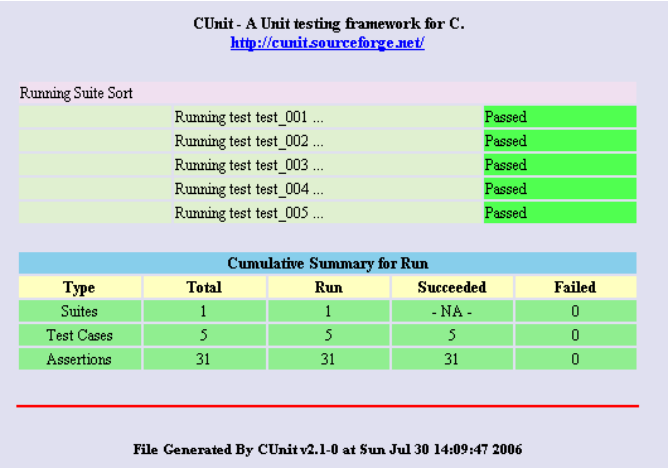
No código fonte são encontrados diversos exemplos que auxiliam o usuário do sistema e oferece suporte a diversas plataformas de desenvolvimento sem fio.

Além dos itens citados acima, a plataforma possui uma comunidade ativa, e oferece um suporte de instalação e utilização eficientes.

IV. TESTES UNITARIOS COM CUNIT

Como já mencionado na introdução, os testes unitários tem a finalidade de testar a menor parte testável do sistema, geralmente um método ou função, e são adicionados ao programa a ser desenvolvido por meio de classes testes. Tem-se como beneficio além da confiabilidade do que se está desenvolvendo, ele garante que problemas serão descobertos cedo ainda durante o processo de desenvolvimento, facilita a manutenção do código por indicar se a unidade ainda está funcional devido algumas mudanças e serve como documentação, oferecendo detalhes da funcionalidade da unidade testada[5].

CUnit[6] é uma plataforma de testes unitários que são executados em linguagem C. Ela utiliza estruturas simples para testes e fornece uma gama possibilidades de teste, além de várias interfaces possíveis de comunicação de resultados, como por exemplo: XML, console interface (ansi C), que são mostrados nas figuras a seguir respectivamente:



CUnit - A Unit testing framework for C.
http://cunit.sourceforge.net/

Running Suite Sort

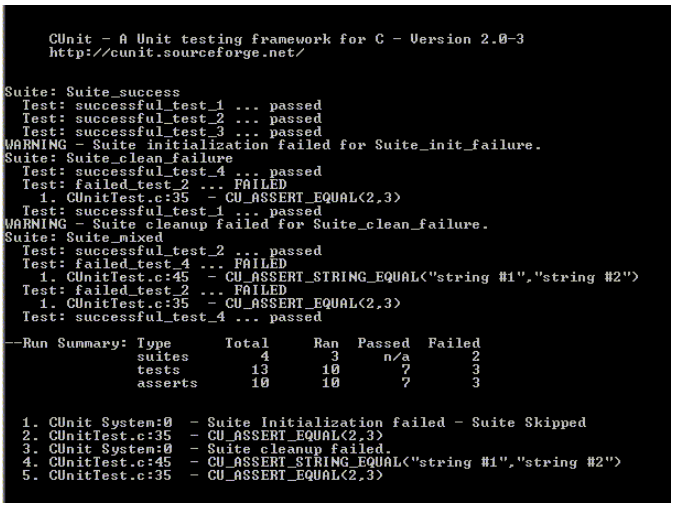
Running test test_001 ...	Passed
Running test test_002 ...	Passed
Running test test_003 ...	Passed
Running test test_004 ...	Passed
Running test test_005 ...	Passed

Cumulative Summary for Run

Type	Total	Run	Succeeded	Failed
Suites	1	1	- NA -	0
Test Cases	5	5	5	0
Assertions	31	31	31	0

File Generated By CUnit v2.1-0 at Sun Jul 30 14:09:47 2006

Fig. 3. Interface XML dos resultados dos testes em CUnit.



```
    CUnit - A Unit testing framework for C - Version 2.0-3
    http://cunit.sourceforge.net/

Suite: Suite_success
  Test: successful_test_1 ... passed
  Test: successful_test_2 ... passed
  Test: successful_test_3 ... passed
WARNING - Suite initialization failed for Suite_init_failure.
Suite: Suite_clean_failure
  Test: successful_test_4 ... passed
  Test: failed_test_2 ... FAILED
    1. CUnitTest.c:35  - CU_ASSERT_EQUAL(2,3)
  Test: successful_test_1 ... passed
WARNING - Suite cleanup failed for Suite_clean_failure.
Suite: Suite_mixed
  Test: successful_test_2 ... passed
  Test: failed_test_4 ... FAILED
    1. CUnitTest.c:45  - CU_ASSERT_STRING_EQUAL("string #1","string #2")
  Test: failed_test_2 ... FAILED
    1. CUnitTest.c:35  - CU_ASSERT_EQUAL(2,3)
  Test: successful_test_4 ... passed

--Run Summary: Type      Total     Ran  Passed  Failed
               suites        4       3     n/a       2
               tests        13      10       7       3
               asserts      10      10       7       3


  1. CUnit System:0  - Suite Initialization failed - Suite Skipped
  2. CUnitTest.c:35  - CU_ASSERT_EQUAL(2,3)
  3. CUnit System:0  - Suite cleanup failed.
  4. CUnitTest.c:45  - CU_ASSERT_STRING_EQUAL("string #1","string #2")
  5. CUnitTest.c:35  - CU_ASSERT_EQUAL(2,3)
```

Fig. 4. Interface Ansi(C) dos resultados dos testes em CUnit[6].

A partir de agora será apresentada uma proposta de integração do RTOS Contiki juntamente com o CUnit, para que em trabalhos futuros seja possível criação de aplicações para placas de baixo custo com garantia de confiabilidade e documentação das funcionalidades do sistema.

V. PROPOSTA DE INTEGRAÇÃO CONTIKI E CUNIT

Primeiramente deve se instalar o CUnit através do *source code*, descompactando-o e configurando corretamente de acordo com os requisitos do sistema.

Para adicionar a ferramenta de testes unitários CUnit ao projeto é necessário iniciar o registro e posteriormente os suítes, que englobam conjuntos de funções responsáveis para testar as partes do projeto. A hierarquia da construção do código é representada na figura a seguir:

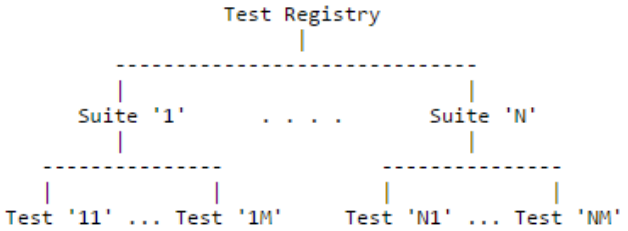


Fig. 5. Estrutura de testes em CUnit[6].

Assim pode se subdividir os testes em setores (Suite ‘1’ a Suite ‘N’) e cada setor possuir várias funções testes (Test ‘11’ a Test ‘NM’).

O escopo das funções depende do que se pretende testar, mas obrigatoriamente devem ser utilizados métodos de comparação e previsão de resultados que são encontrados na biblioteca CUnit.h. Os métodos de teste dessa biblioteca suportam os mais comuns tipos de dados, o que permite uma gama de possibilidades de testes. Posteriormente, se inicializa o registro de teste, onde serão adicionados os suítes e as funções testes criadas no primeiro passo e são adicionadas ao suíte correspondente. O CUnit permite escolher várias interfaces de saída dos resultados, então após os passos anteriores, basta escolher a melhor interface que atende ao problema e gerar os testes. Ao final do programa é necessário limpar o registro de teste.

No sistema operacional Contiki possui diversos exemplos de códigos para diversos tipos de aplicações para plataformas de prototipagem sem fio. E foi utilizado um desses exemplos para integração do Contiki com o CUnit. O ápice do trabalho se deu em adicionar as dependências do CUnit ao projeto Contiki. No arquivo de compilação Makefile do Contiki foi adicionado a biblioteca estática do CUnit posteriormente foi realizado o “*link*” dos objetos criados pelos dois sistemas no objetivo de gerar o executável final contemplando a integração.

Com essa integração é possível criar um ambiente de desenvolvimento de aplicações totalmente respaldado à garantia de funcionamento através dos testes unitários.

VI. ESTUDO DE CASO

Para realização da prova de conceito, adicionou-se o suporte à utilização do CUnit dentro do ambiente do Contiki, e realizou-se testes unitários alterando-se os argumentos de funções presentes nos módulos /core/net/ipv6, de tal forma a realizar testes que produzem sucesso e também falhas, por meio da variação dos parâmetros enviados como argumentos para as funções.

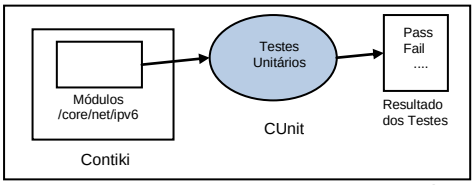


Fig. 5. Estudo de caso – Integração e Testes Unitários

Por meio da integração realizada e dos testes efetuados, validou-se a ideia da integração proposta no presente trabalho.

VII. CONCLUSÕES

As redes de sensores sem fio se mostram eficientes quanto a confiabilidade, facilidade de instalação, flexibilidade e capacidade de operar em locais que redes com fio são inviáveis [7].

Com este trabalho de integração de um sistema operacional com uma plataforma de testes de unidade abre o leque de possibilidades de se desenvolver aplicações robustas e que podem ser implementadas no ramo de redes industriais.

Visando isto, trabalhos futuros serão destinados à criação de redes de sensores sem fio utilizando plataformas de prototipação e rádios transmissores de baixo custo, onde será possível o respaldo visível de falhas e garantia de funcionamento.

VIII. REFERÊNCIAS

[1] Embarcados (2016). *Sistemas Operacionais de Tempo Real.* Acedido em 31 de Maio de 2016, em: http://www.embarcados.com.br.

[2] Embarcados (2016). *Embarcados: Desenvolvendo um RTOS.* Acedido em 03 de Junho de 2016, em http://www.embarcados.com.br.

[3] Contiki. *The Open Source OS for the Internet of Things.* Acedido em 01 de Junho de 2016, em: http://www.contiki-os.org.

[4] RS-Online (2016). *RS-Online DesignSpark*. Acedido em 04 de Junho de 2016, em http://www.rs-online.com

[5] B. Kent, "Test-Driven Development By Example", *Three Rivers Institute*, 1ª Edição, 2003.

[6] CUnit Home (2016). *CUnit A Unit Testing Framework for C.* Acedido em 03 de Junho de 2016, em: cunit.sourceforge.net.

[7] A. A de Souza, H. A. Castro, R. S. Carrijo, "Proposta de uma solução alternatica aplicada em rede de sensors sem fio no padrão IEEE 802.15.4", CEEL 2015, Outubro/2015.