Projeto de Automação de Teste de Software em Sistemas Embarcados com ESP32 para Monitoramento de Qualidade da Água.

Alom Estevam de Paiva

Objetivo

O projeto tem como objetivo analisar e propor soluções para a automação de tarefas ligadas a teste de software em sistemas embarcados, com foco no ESP32, aplicado ao monitoramento de qualidade da água. O escopo inclui o estudo de abordagens para planejar, criar e manter testes automáticos que verifiquem a confiabilidade e eficiência de tais sistemas. A motivação principal é melhorar a eficiência dos processos de teste e assegurar que sistemas embarcados funcionem adequadamente em condições críticas.

Escopo

Tipos, fases e abordagens de teste

Tipos de Testes: Unitário, e de integração.

Fases:

Planejamento: Definir a estratégia de teste para cada componente (placa, e sensores).

Execução: Testar leituras dos sensores, transmissão de dados, funcionamento da interface de monitoramento.

Monitoramento e Controle: Utilizar a interface web para acompanhar a execução dos testes em tempo real.

Teste Estático, Estrutural e Funcional

Teste Estático: Revisão do código que controla os sensores.

Teste Estrutural: Avaliação da arquitetura da placa eletrônica embarcada para garantir que o fluxo de dados seja eficiente e confiável.

Teste Funcional: Validação do sistema de monitoramento da boia de captação de dados, garantindo que as leituras (temperatura, pH, oxigênio) sejam precisas

---

Motivação para a Pesquisa

Este projeto foca na criação de uma solução de automação de testes para um sistema de monitoramento de qualidade da água em viveiros de peixes. Usando ferramentas embarcadas e sensores conectados a uma interface de

monitoramento, como demonstrado nas imagens, o objetivo é validar a precisão e eficiência do sistema IoT de monitoramento de água, garantindo que ele funcione de forma eficiente em tempo real. Ainda tive um grande desafio em encontra artigos e trabalhos que abrangesse todas as palavras chaves do tema de meu projeto, e que tivesse aplicações relevantes e correlatas e que pudesse utiliza-las em meu projeto, no entanto foi adquirido um apanhado de modelos e ferramentas que proporcionaram meios de suporte para elucidação de nosso projeto.

Strings de Busca

- "Automated testing in embedded systems"
- "Testes automatizados ESP32 aquicultura"
- "Integration testing hardware/software"
- "Test strategies IoT devices"
- "Automated frameworks for embedded software"

Listagem do Material Encontrado e Comentários

1. **Referência:** “Estratégia de teste de integração de hardware/software automatizado enxuto para sistemas embarcados” Floriano Muttenthaler ;Stefan Wilker ;Thilo Sauter2021 22ª Conferência Internacional IEEE sobre Tecnologia Industrial (IC/IT) Ano: 2021 | Artigo de conferência |Editora: IEEE

   Nota: Apresenta uma estratégia moderna e eficiente para integrar testes automáticos em sistemas embarcados, destacando o uso de pipelines CI/CD e automação enxuta.

   “Resumo:

   A integração de componentes de software em sistemas críticos de segurança requer verificação dedicada de componentes de software na plataforma de hardware de destino. Executar testes Processor In the Loop (PIL) em um microcontrolador como verificação de componentes de software em um ambiente de destino é estabelecido como um método comum. Para a verificação automatizada de periféricos de hardware associados, a estratégia de teste PIL é estendida pela aplicação de testes abrangentes de Hardware In the Loop (HIL). A verificação funcional com base em uma configuração de teste PIL requer instrumentação extensiva além do software produtivo no microcontrolador. Uma estratégia alternativa para a automação de métodos de verificação de integração de hardware/software em ambientes de destino de microcontrolador é proposta neste trabalho. Ao realizar testes na plataforma host que, por um lado, interage com o sistema de teste HIL e, por outro lado, se comunica com o microcontrolador por meio de um depurador, supera a instrumentação extensiva no ambiente de destino do microcontrolador. Um conceito para uma plataforma de teste e como usá-la para testes de componentes de software e integração, bem como para testes de integração de hardware/software em um determinado

ambiente de destino é apresentado neste artigo. Esta plataforma foi avaliada verificando componentes de software Low-Level Driver no respectivo ambiente de destino do microcontrolador. A eficiência do tempo de execução do teste com execução automatizada foi comparada e analisada com aquelas dos testes manuais.”

2. **Referência:** V. Garousi, M. Felderer, Ç. M. Karapıçak e U. Yılmaz, "O que sabemos sobre testes de software embarcado", em IEEE Software , vol. 35, no. 4, pp. 62-69, julho/agosto de 2018, doi: 10.1109/MS.2018.2801541

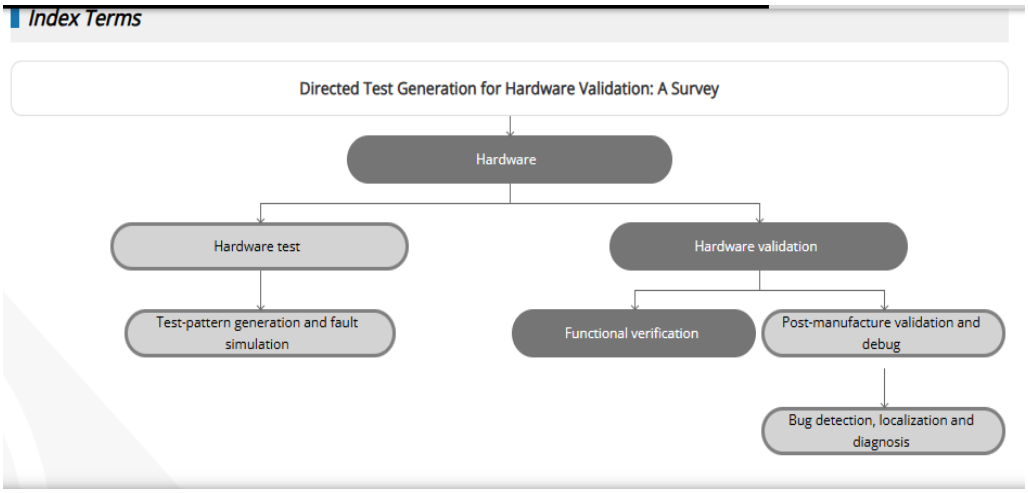
   Nota: Este estudo oferece um panorama abrangente sobre testes de software embarcado. Ele pode complementar a revisão de literatura ao detalhar práticas, desafios e tendências relevantes para sistemas embarcados, fornecendo base teórica sólida para o seu trabalho.

   “Resumo:

   Para testar software embarcado de forma econômica, profissionais e pesquisadores propuseram muitas técnicas de teste, abordagens, ferramentas e estruturas. No entanto, obter uma visão geral do estado da arte e do estado da prática nessa área é desafiador para profissionais ou novos pesquisadores. Além disso, devido a uma visão geral inadequada do que já existe nessa área, algumas empresas geralmente reinventam a roda ao projetar uma abordagem de teste que é nova para elas, mas já existe. Para abordar esses problemas, os autores conduziram uma revisão sistemática da literatura dessa área que cobriu os tópicos de teste, atividades de teste, artefatos de teste e indústrias nas quais os estudos se concentraram. Os resultados podem beneficiar tanto profissionais quanto pesquisadores, servindo como um índice para o vasto corpo de conhecimento nessa área importante e de rápido crescimento.”

3. **Referência:** Directed Test Generation for Hardware Validation: A Survey Crossref DOI link: https://doi.org/10.1145/3638046 Published Online: 2024-01-12 Published Print: 2024-05-31 Update policy: https://doi.org/10.1145/crossmark-policy
   Nota: Este levantamento explora técnicas de geração de testes direcionados para validação de hardware. Pode ser especialmente útil para o seu projeto ao propor estratégias que assegurem a compatibilidade e funcionalidade do hardware com os requisitos do sistema.
   “Resumo
   A complexidade dos projetos de hardware aumentou ao longo dos anos devido ao rápido avanço da tecnologia, juntamente com a necessidade de oferecer suporte a recursos diversos e complexos. A crescente complexidade do projeto se traduz diretamente na dificuldade de verificação de comportamentos funcionais, bem como de requisitos não funcionais. A simulação é a forma de validação mais amplamente usada, usando padrões de teste aleatórios e aleatórios restritos. A natureza aleatória das sequências de teste pode cobrir a grande maioria dos cenários; no entanto, pode introduzir uma sobrecarga inaceitável para cobrir todos os cenários funcionais e não funcionais possíveis. Os testes direcionados são promissores para cobrir os casos restantes e os

cenários difíceis de detectar. O desenvolvimento manual de testes direcionados pode ser demorado e sujeito a erros. Um caminho promissor é realizar a geração automatizada de testes direcionados. Neste artigo, fornecemos um levantamento abrangente de técnicas de geração de testes direcionados para validação de hardware.



Especificamente, primeiro apresentamos a complexidade da verificação de hardware para destacar a necessidade de geração de testes direcionados. A seguir, descrevemos a geração de testes direcionados usando diversas técnicas automatizadas, incluindo métodos formais, testes concólicos e aprendizado de máquina. Por fim, discutimos como utilizar efetivamente os padrões de teste gerados em diferentes cenários de validação, incluindo validação funcional pré-silício, depuração pós-silício, bem como validação de requisitos não funcionais."

4. **Referência:** Mulhem S Ewert C Nešković A Poudel A(2024) Testes de campo híbridos de software/hardware seguros para sistema em chip 2024 IFIP/IEEE 32ª Conferência Internacional sobre Integração em Grande Escala (VLSI-SoC) 10.1109/VLSI-SoC62099.2024.10767817 (1-6) Data de publicação online: 6-out-2024 https://doi.org/10.1109/VLSI-SoC62099.2024.10767817

   Nota: Este artigo aborda testes híbridos de software/hardware voltados para sistemas em chip, com ênfase em segurança. Ele pode fornecer insights valiosos sobre como melhorar a confiabilidade de sistemas embarcados e IoT aplicados ao ESP32. Seria útil explorar as metodologias de teste apresentadas e como elas podem ser adaptadas ao seu projeto.

   "Resumo:

   Os modernos sistemas em chip (SoCs) incorporam módulos de autoteste (BIST) integrados profundamente nos blocos de propriedade intelectual (IP) do dispositivo. Esses módulos lidam com falhas e defeitos de hardware durante a operação do dispositivo. Como tal, os resultados do BIST potencialmente revelam a estrutura interna e o estado do dispositivo em teste (DUT) e, portanto, abrem vetores de ataque. A chamada compactação de resultados pode superar essa vulnerabilidade ocultando a estrutura da cadeia BIST, mas introduz os problemas de aliasing e assinaturas inválidas. O software-BIST fornece uma solução flexível, que pode lidar com esses problemas, mas sofre de

observabilidade limitada e cobertura de falhas. Neste artigo, portanto, apresentamos uma abordagem híbrida de software/hardware de baixa sobrecarga que supera as limitações mencionadas. Ela depende de (um) código de autenticação de mensagem de hash com chave (KMAC) disponível no S oC, fornecendo assinaturas seguras e válidas específicas do dispositivo com aliasing zero e (b) o processador S oC para agendamento de testes, aumentando assim a disponibilidade do DUT. A abordagem proposta oferece recursos de teste no chip e remoto. Apresentamos um S oC baseado em RISC-V para demonstrar nossa abordagem, discutindo a sobrecarga do sistema e as taxas de compactação resultantes."

5. **Referência:** "Testes automatizados e o hardware". Embárcados.

   Nota: Discussão sobre a importância de testes automatizados para sistemas embarcados e IoT. Destaque para o uso de ferramentas como Pytest e Unity Framework.

   "Introdução

   Queremos portanto validar que para um dado cenário de execução da função, ou seja, para uma situação em que as variáveis das quais a função depende estejam em um valor especificado pelo teste, ações serão tomadas corretamente, e a saída da função será a esperada.

   Em sistemas embarcados muitas vezes o problema reside na utilização de um periférico do microcontrolador que não está presente na máquina de desenvolvimento, então não é possível testar aquele código, certo? Errado!

   Primeiro o objetivo do seu teste não deveria ser: O SPI está funcionando, mas sim, estou mandando as informações corretas ao SPI. O primeiro teste é sim relevante e só pode ocorrer com o uso do dispositivo e no hardware em questão. Entretanto, o segundo cenário é igualmente relevante e muito frequente.

   Motivos para usar um substituto no seu teste

   Antes de chegar ao uso das bibliotecas que pretendo apresentar neste texto, quero apontar alguns motivos principais para o uso de um substituto:

   1 – É uma dependência que não faz parte do objetivo do meu teste.

   Quando estamos escrevendo um teste para um dado subsistema, temos a tendência de querer chamar todas as suas dependências. Isso não é necessário em todos os estágios. Vamos considerar o exemplo de uma aplicação em IoT que faz uso de um modem 3G para a comunicação. Se vamos testar a montagem e envio das requisições precisamos do código referente ao modem, correto? Errado! O que precisamos é determinar como será a interface para o código cliente que utilizará o modem e definir o "contrato" de utilização. Naturalmente é importante ter testes que integrem as duas soluções de fato, mas esse é um outro teste em uma outra fase. Um passo de cada vez.

2 – É uma dependência que é impossível de ser integrada no ambiente de desenvolvimento.

Esse é o caso típico de quando vamos acessar um periférico em um microcontrolador. O endereço de memória em que ele está mapeado não pode ser acessado no ambiente de desenvolvimento pois não há o periférico por lá.

3 – Existe um tempo razoável para o uso do recurso.

Consideremos novamente a aplicação IoT que deve trocar mensagem com um sistema na nuvem. Pro seu código, que vai tratar a informação, não é relevante saber que a conexão foi fechada via TLS e o servidor respondeu em um determinado tempo. O que é relevante é: Enviei a requisição X, recebi a resposta Y e tomei a ação Z.

Resumindo e voltando a questão do objetivo do teste: Queremos validar o comportamento do código em teste.

Um caso de teste: Aguardando um semáforo e enviando informação via SPI.

Para mostrar situações que são típicas ao escrever testes em sistemas embarcados vamos tomar como exemplo uma aplicação que aguarda um semáforo para enviar via SPI um valor, e em caso de timeout envia um valor diferente. É um caso de teste simples mas nos permite propor algumas ideias:

Como lidar com o loop infinito que aparece nas funções que executam threads em RTOS?

Como controlar uma dependência que faz parte do meu teste para gerar casos de teste?

Como tratar o acesso ao hardware?

Uma pausa para escrita do teste: Catch

Para a escrita do teste vamos usar a ferramenta chamada Catch. Há outros frameworks de teste disponíveis e você pode experimentar para encontrar aquele que lhe agrade mais. O nosso primeiro passo é descrever o comportamento do nosso sistema.

```
SCENARIO("Uma informação deve ser enviada via SPI"){

  GIVEN("A requisição de envio virá antes do timeout"){

    THEN("O sistema enviará o valor 0xCA") {

    }

  }
```

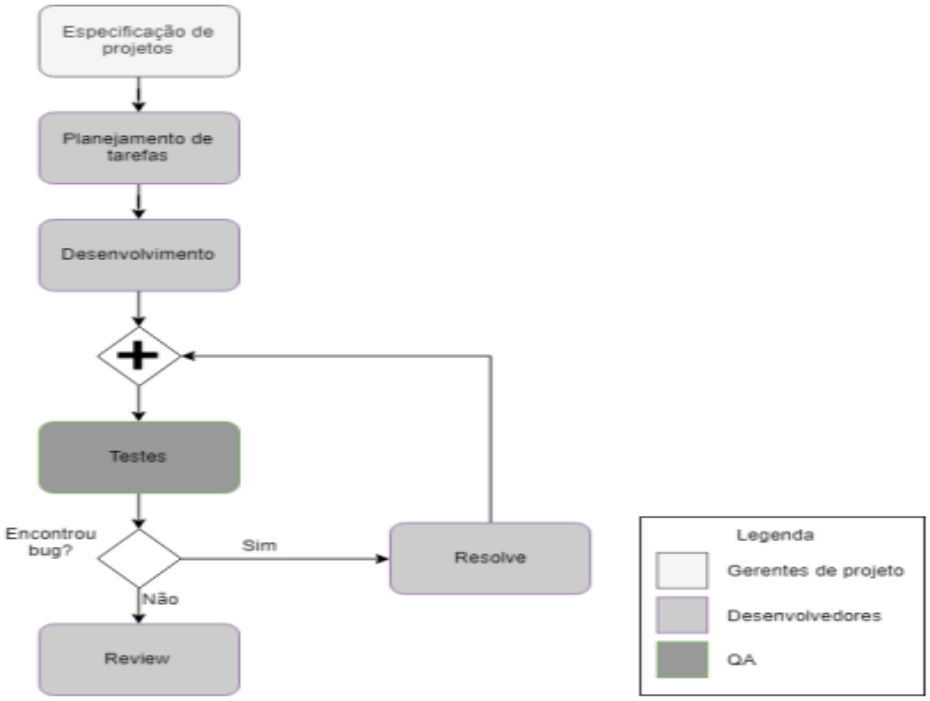
```
 GIVEN("Haverá o timeout antes da requisição de envio"){

  THEN("O sistema enviará o valor 0xFE") {

  }

 }

}”
```

6. **Referência:** PFC de Rafael Figueiró Berto. "Proposta de Framework para Testes de Software Embarcado". UFSC, 2019.
   Nota: Trabalho acadêmico que apresenta um framework para facilitar a automação de testes em software embarcado. Aplicação direta ao ESP32 e cenários de IoT.

   “Resumo

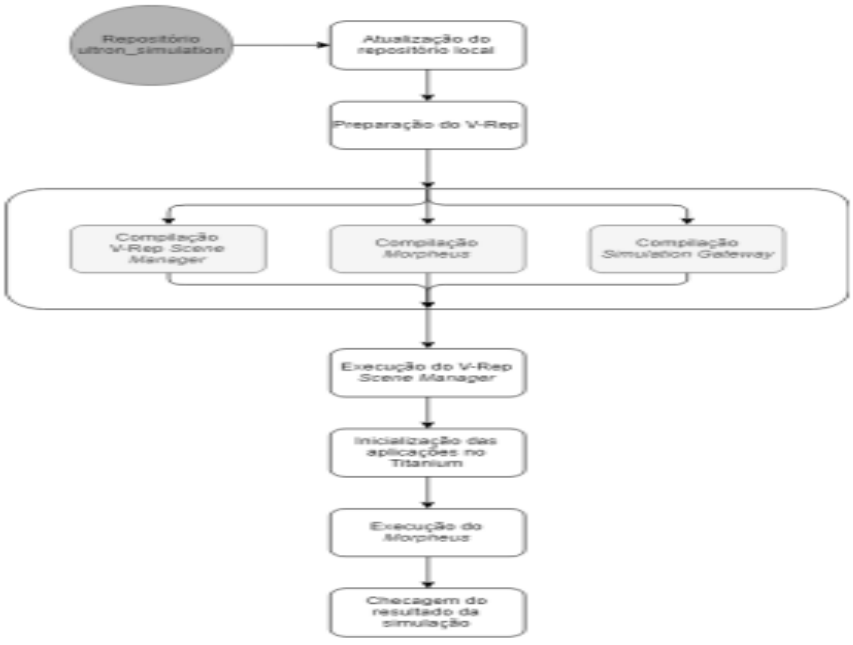
   A Hexagon Agriculture é uma empresa que desenvolve soluções para digitalização da agricultura, combinando tecnologias em hardware, software e know-how para converter dados em informações inteligentes e acionáveis e trazendo soluções para cultivo, colheita e Original Equipment Manufacturer (OEM). No cotidiano do desenvolvimento é necessária a validação de cada funcionalidade antes dela ser entregue ao cliente, assim, evitando danos a imagem da empresa e do produto, para isso são realizados diversos testes do software e do hardware produzidos. A realização destes testes ocorre em duas etapas: testes unitários, realizados pelos desenvolvedores, e testes de usabilidade, realizados pela equipe de garantia de qualidade (QA).

Figura 1 – Processo sistêmico - P&D.



Fonte: Adaptado de [18, 19].

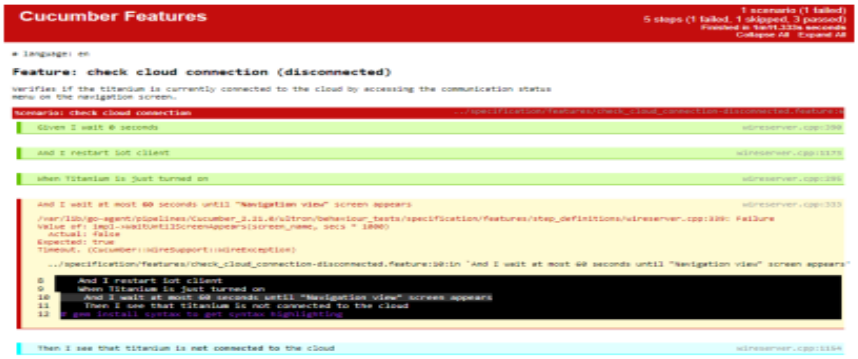
Figura 9 – Diagrama da simulação do piloto automático.



Fonte: Autor.

Os testes realizados pelo QA possibilitam uma visão do produto funcionando como um todo, porém exigem uma elevada quantidade de maquinário, espaço físico, mão-de-obra e tempo. Buscando localizar problemas relacionados ao desenvolvimento, de forma mais rápida e sem a necessidade da elevada quantidade de recursos, este trabalho apresenta o desenvolvimento de técnicas de simulação e teste de software automatizada, tendo como objetivo realizar testes e simulações diariamente, ainda na fase de desenvolvimento.

ANEXO B – Exemplo de resultado dos testes de comportamento.



Fonte: Resultado de teste de sistema desenvolvido pela *Hexagon Agriculture* através do *framework* Cucumber.

Com o auxílio destas ferramentas se espera a melhora na qualidade dos produtos, redução do tempo gasto em testes e na solução de problemas.”

7. **Referência:** MAJ Burford e F. Belli. "CADAS: Uma ferramenta para projetar software embarcado confiável". Springer, 1984.
   Nota: apesar da idade que é mais antiga do que eu mais não dispensa Discussão sobre o CADAS, uma ferramenta clássica voltada para a confiabilidade de software embarcado, Aplica-se bem ao contexto histórico de teste de sistemas críticos. Porém não conseguir ter acesso.

   “Resumo

   O desenvolvimento de sistemas de software embarcados tem sido atormentado por problemas de confiabilidade, disponibilidade e altos custos. O desenvolvimento do ADA reconheceu a contribuição da linguagem de implementação para esse déficit orçamentário. Infelizmente, o do ambiente de suporte, incluindo verificação e validação e a interação de seus componentes individuais, ainda precisa ser geralmente reconhecido e compreendido.

   Para definir uma ferramenta ideal para verificação e validação, sua interação com outros componentes do ambiente de suporte de programação deve ser reconhecida primeiro. O design e a realização parcial de tal ferramenta foram alcançados por meio do desenvolvimento do CADAS (Computer Aided Design And Verification System). Neste artigo, seu desenvolvimento, implementação e uso até o momento são discutidos. Um conceito de como ele pode ser usado para dar suporte ao desenvolvimento e verificação de manipuladores de exceção é apresentado.”

8. **Referência:** A. Banerjee, S. Chattopadhyay e A. Roychoudhury. "Sobre testes de software embarcado". Advances in Computers, 2016.

Nota: Explora as especificidades dos testes em sistemas embarcados, incluindo as dificuldades de reproduzir cenários reais e as limitações de hardware.

“Resumo

Nas últimas décadas, os sistemas embarcados expandiram seu alcance para os principais aspectos da vida humana. Começando com pequenos dispositivos portáteis (como smartphones) até sistemas automotivos avançados (como sistemas de freios antibloqueio), o uso de sistemas embarcados aumentou em um ritmo dramático. Software embarcado é um software especializado que se destina a operar em dispositivos embarcados. Neste capítulo, descreveremos os desafios exclusivos associados ao teste de software embarcado. Em particular, o software embarcado é necessário para satisfazer várias restrições não funcionais, além de restrições relacionadas à funcionalidade. Essas restrições não funcionais podem incluir (mas não se limitar a), restrições relacionadas ao tempo/consumo de energia ou requisitos de confiabilidade, etc. Além disso, os sistemas embarcados geralmente são necessários para operar em interação com o ambiente físico, obtendo suas entradas de fatores ambientais (como temperatura ou pressão do ar). A necessidade de interagir com um ambiente físico dinâmico, geralmente não determinístico, aumenta ainda mais os desafios associados ao teste e à validação de software embarcado. No passado, as metodologias de teste e validação foram estudadas extensivamente. Este capítulo, no entanto, explora os avanços nas metodologias de teste de software, especificamente no contexto de software embarcado. Este capítulo apresenta ao leitor os principais desafios em testar propriedades não funcionais de software por meio de exemplos realistas. Ele também apresenta uma classificação fácil de seguir de trabalhos de pesquisa existentes sobre este tópico. Finalmente, o capítulo é concluído com uma revisão de direções futuras promissoras na área de testes de software embarcado.”

9. **Referência:** C. Ebert e C. Jones. "Software embarcado: fatos, números e futuro". IEEE, 2009.
   Nota: Relato abrangente sobre o estado da arte de software embarcado, destacando a evolução de testes e integração com processos de engenharia.

   “Resumo:

   Devido ao contexto complexo do sistema de aplicativos de software embarcado, defeitos podem causar situações de risco de vida, atrasos podem criar custos enormes e produtividade insuficiente pode impactar economias inteiras. Fornecer melhores estimativas, definir objetivos e identificar pontos críticos na engenharia de software embarcado requer dados de benchmarking adequados.”

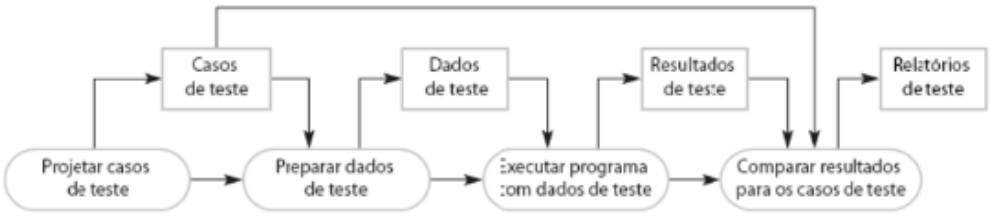
10. **Referência:** SILVA, Mario Luis Moreira da; DALLILO, Felipe Diniz. "Uma visão geral sobre automação de testes". Revista Científica Multidisciplinar Núcleo do Conhecimento, 2019.
    Nota: O artigo fornece uma visão geral sobre automação de testes, destacando ferramentas e boas práticas. Apresenta conceitos aplicáveis à automação de testes em sistemas embarcados.

“Resumo

As organizações responsáveis pelo desenvolvimento de softwares, no mundo empresarial atual, buscam por técnicas e estratégias inovadoras tendo-se como escopo o aperfeiçoamento dos seus processos bem como a entrega de produtos em um período de tempo apontado pelos consumidores como adequado, para, assim, atender o que almejam esses que consomem, visto que, atualmente, são menos tolerantes à erros e são bastante exigentes frente ao que adquirem. Software em desenvolvimento requer um grande esforço em testes minuciosos por estarem mudando constantemente e através dos testes automatizados é possível obter um feedback mais rápido, por exemplo, de visualizar quais testes passaram e quais falharam, qual passo teve erro e qual o motivo.

Figura 2: Modelo do teste

Figura 2 Modelo do Processo de Teste de Software



Fonte: SOMMERVILLE(2011)

Durante o desenvolvimento, é necessário tomar a decisão de automatizar ou não os cenários de testes, para garantir que não há regressões no software e economizar tempo. Este artigo avalia de forma descritiva requisitos de automação de testes, mostrando à importância nas etapas de produção e execução, discorrendo, nesse processo, sobre as teorias que orientam a realização de atividades de teste, as técnicas e critérios. Escolheu-se por aqueles que podem ser acionados tanto para verificar quanto para validar um produto.”

Conclusão

A revisão do estado da arte revelou uma diversidade de ferramentas, técnicas e abordagens para automação de testes em sistemas embarcados. Destaques:

- **Desafios principais:** Recursos limitados de hardware, dificuldade na reprodução de condições reais e falta de integração com ferramentas de CI/CD.
- **Ferramentas útéis:** Pytest, Unity Framework e Espressif IDF Testing Framework.
- **Tendências:** Uso de estratégias enxutas e integração com pipelines automatizados para maior eficiência.

Sugere-se que futuros estudos explorem soluções específicas para integração de sensores e simulação de cenários reais, aumentando a confiabilidade dos sistemas testados.

Referências

1. Muttenthaler, Florian, et al. Estratégia de teste de integração de hardware/software automatizado enxuto para sistemas embarcados. IEEE, 2021.
2. V. Garousi, M. Felderer, Ç. M. Karapıçak e U. Yılmaz, "O que sabemos sobre testes de software embarcado", em IEEE Software , vol. 35, no. 4, pp. 62-69, julho/agosto de 2018, doi: 10.1109/MS.2018.2801541
3. Directed Test Generation for Hardware Validation: A Survey Crossref DOI link: https://doi.org/10.1145/3638046 Published Online: 2024-01-12 Published Print: 2024-05-31 Update policy: https://doi.org/10.1145/crossmark-policy
4. Mulhem S Ewert C Nešković A Poudel A(2024) Testes de campo híbridos de software/hardware seguros para sistema em chip 2024 IFIP/IEEE 32ª Conferência Internacional sobre Integração em Grande Escala (VLSI-SoC) 10.1109/VLSI-SoC62099.2024.10767817 (1-6) Data de publicação online: 6-out-2024 https://doi.org/10.1109/VLSI-SoC62099.2024.10767817
5. "Testes automatizados e o hardware". Embárcados. Disponível em: Embárcados.
6. Berto, Rafael Figueiró. Proposta de Framework para Testes de Software Embarcado. UFSC, 2019. Disponível em: Repositório UFSC.
7. Burford, MAJ e Belli, F. CADAS: Uma ferramenta para projetar software embarcado confiável. Springer, 1984.
8. Banerjee, A., Chattopadhyay, S., e Roychoudhury, A. Sobre testes de software embarcado. Advances in Computers, 2016.
9. Ebert, C. e Jones, C. Software embarcado: fatos, números e futuro. IEEE, 2009.
10. SILVA, Mario Luis Moreira da; DALLILO, Felipe Diniz. Uma visão geral sobre automação de testes. Revista Científica Multidisciplinar Núcleo do Conhecimento. Dezembro de 2019. Disponível em: Núcleo do Conhecimento.

ANEXOS

ANEXOS

TRABALHOS FUTUROS
Extensão do Projeto para testes e implementação em um aplicativo Mobile criado e ainda com o software em construção:
A configuração do Dart/Flutter, listando pacotes instalados e suas versões. Com base nele, podemos identificar possíveis automações de teste de software implementadas no projeto, considerando bibliotecas específicas relacionadas a testes e cobertura de código.

Automação de Testes Unitários e de Integração:

flutter_test: Biblioteca padrão para testes em projetos Flutter. Suporte a:
Testes unitários para verificar funcionalidades específicas de métodos.
Testes widget para interações entre componentes.
mockito (não listado, mas frequentemente usado): Para criar mocks e isolar dependências.
Cobertura de Código:
coverage: Ferramenta para calcular cobertura de testes, útil para medir o percentual de código testado.

Teste automatizado de interface gráfica (UI) em Flutter, mais especificamente um teste de widget. Esse tipo de teste é usado para verificar o comportamento e as interações de widgets individuais na interface de uma aplicação Flutter.

Características do teste:
Teste de Widget:
O código utiliza a função testWidgets, que é específica para testar widgets.
Ele simula a interação com a interface do usuário, como cliques e atualizações de estado.
Simulação de interação:

Utiliza o WidgetTester para manipular widgets:
pumpWidget: Carrega o widget principal da aplicação (MyApp).
find: Localiza elementos na árvore de widgets, como texto ou ícones.
tap: Simula o toque em um ícone (Icons.add).
Verificação de estados:
Usa expect para validar o estado do widget antes e depois da interação:
Antes da interação: O contador exibe "0".
Após o toque no botão: O contador exibe "1".
Classificação do teste:
Esse é um exemplo de teste funcional automatizado aplicado à interface, com foco em garantir que os widgets do aplicativo funcionem corretamente em diferentes cenários. Especificamente, pode ser considerado um smoke test (teste de fumaça), pois verifica rapidamente se a funcionalidade básica de incremento do contador está funcionando.

```
// This is a basic Flutter widget test.
//
// To perform an interaction with a widget in your test, use the WidgetTester
// utility in the flutter_test package. For example, you can send tap and scroll
// gestures. You can also use WidgetTester to find child widgets in the widget
// tree, read text, and verify that the values of widget properties are correct.
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';
import 'package:solaris/main.dart';
void main() {
  testWidgets('Counter increments smoke test', (WidgetTester tester) async {
    // Build our app and trigger a frame.
    await tester.pumpWidget(MyApp());
// Verify that our counter starts at 0.
    expect(find.text('0'), findsOneWidget);
    expect(find.text('1'), findsNothing);
// Tap the '+' icon and trigger a frame.
    await tester.tap(find.byIcon(Icons.add));
    await tester.pump();
// Verify that our counter has incremented.
    expect(find.text('0'), findsNothing);
    expect(find.text('1'), findsOneWidget);
  });
}
```

Projeto de Automação de Teste de Software em Sistemas Embarcados com ESP32 para Monitoramento de Qualidade da Água.

Componente Curricular: Teste de Software
Professor: Sidney Nogueira
Professor orientador: Victor Medeiros; Obionor Nobrega
Aluno: Alom Estevam

Visão Geral

Este projeto foca na criação de uma solução de automação de testes para um sistema de monitoramento de qualidade da água em viveiros de peixes. Usando ferramentas embarcadas e sensores conectados a uma interface de monitoramento, como demonstrado nas imagens, o objetivo é validar a precisão e eficiência do sistema IoT de monitoramento de água, garantindo que ele funcione de forma eficiente em tempo real.

Objetivos

- Testar e validar a **placa eletrônica** equipada com ESP32 e outros sensores para garantir a eficiência no monitoramento da qualidade da água.
- Utilizar o **Power Profiler Kit II** para verificar o consumo de energia dos componentes eletrônicos em diferentes condições de operação.
- Implementar testes unitários e funcionais automatizados para garantir a confiabilidade dos dados coletados pelos sensores em um viveiro de peixes, como a boia de captação de qualidade da água.

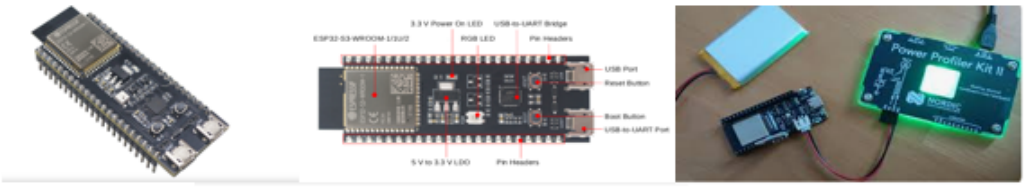


Figura 1: ESP32-S3-DevKitC-1 v1.1 Componentes, Power Profiler Kit II

VISÃO GERAL DE TESTE DE SOFTWARE

Tipos, fases e abordagens de teste

- **Tipos de Testes**: Unitário, e de integração.
- **Fases**:
  - **Planejamento**: Definir a estratégia de teste para cada componente (placa, e sensores).
  - **Execução**: Testar leituras dos sensores, transmissão de dados, funcionamento da interface de monitoramento.
  - **Monitoramento e Controle**: Utilizar a interface web para acompanhar a execução dos testes em tempo real.

Teste Estático, Estrutural e Funcional

- **Teste Estático**: Revisão do código que controla os sensores.
- **Teste Estrutural**: Avaliação da arquitetura da placa eletrônica embarcada para garantir que o fluxo de dados seja eficiente e confiável.
- **Teste Funcional**: Validação do sistema de monitoramento da boia de captação de dados, garantindo que as leituras (temperatura, pH, oxigênio) sejam precisas.

Planejamento, Monitoramento e Controle dos Testes

- **Planejamento**: Realizar testes automáticos para diferentes componentes: sensor de oxigênio, boia de captação, sistema de alerta.
- **Monitoramento**: Acompanhamento contínuo com o uso da interface de software, permitindo a visualização dos dados em tempo real, como visto na imagem do Power Profiler Kit II.
- **Controle**: Ajustes nos parâmetros de teste baseados nas condições ambientais e consumo de energia.

Automação de Testes de Sistema

A automação será implementada para:

- Testes unitários de módulos do sistema (sensores e interface de comunicação).
- Testes funcionais envolvendo a transmissão de dados em tempo real.
- Testes de integração para validar a interação entre diferentes partes do sistema, como a boia de captação de dados.

METODOLOGIA

Focar na automação de testes em um ambiente de aquicultura com monitoramento contínuo da qualidade da água. O objetivo é garantir que o sistema IoT funcione com eficiência energética e confiabilidade de dados.

Ferramentas Utilizadas:

- **ESP32**: Controle dos sensores e comunicação com a interface de monitoramento.
- **Power Profiler Kit II**: Medição precisa do consumo de energia dos sensores e módulos de comunicação.
- **Software de Monitoramento**: Interface gráfica para acompanhar os testes e a coleta de dados dos sensores de qualidade da água.

Figura 2: protótipo da placa IoT captadora dos dados, acomodado em boias

Automação de Testes

Aplicação de Testes:

1. **Testes Unitários**: Foco nos módulos de sensores de qualidade da água.

Figura 3: sensores de captação de temperatura e PH da água

Figura 4: (a) Arquitetura de rede LoRaWAN; e (b) pilha de protocolos LoRaWAN.

2. **Testes Funcionais**: Verificação da transmissão de dados e alertas gerados pela boia de captação.

3. **Testes de Integração**: Integração entre os sensores, interface web e sistema de controle do aerador para manter o ambiente aquático ideal.

Figura 5: **Power Profiler Kit II, analise de consumo eficiência energética**

CONCLUSÃO

Este projeto é uma aplicação prática da automação de testes para sistemas IoT no contexto da Aquicultura 4.0, utilizando ferramentas de medição de energia e monitoramento contínuo, viabilizando um fornecimento de eficiência energética autônomo, e garantia de consumo dimensionado.

Referências:

- (2021). Datasheet ESP32 Series. Espressif Systems. Version 3.5.
- (2022). Datasheet BME280. Bosch. Rev. 1.23.
- Bouguera, T., Diouris, J.-F., Chaillout, J.-J., Jaouadi, R., and Andrieux, G. (2018). Energy consumption model for sensor nodes based on LoRa and LoRaWAN. 18(7):2104.
- Brunelli, D., Moser, C., Thiele, L., and Benini, L. (2009). Design of a solar-harvesting circuit for batteryless embedded systems. IEEE Transactions on Circuits and Systems I: Regular Papers, 56(11):2519–2528. 2.
- Muttenthaler, Florian, et al. Estratégia de teste de integração de hardware/software automatizado enxuto para sistemas embarcados. IEEE, 2021.
- V. Garousi, M. Felderer, Ç. M. Karapıçak e U. Yılmaz, "O que sabemos sobre testes de software embarcado", em IEEE Software , vol. 35, no. 4, pp. 62-69, julho/agosto de 2018, doi: 10.1109/MS.2018.2801541
- Directed Test Generation for Hardware Validation: A Survey Crossref DOI link: https://doi.org/10.1145/3638046 Published Online: 2024-01-12 Published Print: 2024-05-31 Update policy: https://doi.org/10.1145/crossmark-policy