Universidade Federal de Roraima
Departamento de Ciência da Computação
Introdução a Sistemas Embarcados

DISCIPLINA: INTRODUÇÃO A SISTEMAS EMBARCADOS
ALUNO: Talles Bezerra de Assunção

Lista de Exercício 02

1. Sistema que possui um display de 7 segmentos que exibe um número de 0 até 9 e dois botões, o da esquerda que incrementa ou decrementa o valor no display e o da direita que inverte a contagem de crescente para decrescente ou vice-versa.
Link: https://circuits.io/circuits/3998217-contador-display

2. Sistema que simula um sistema de elevador, possui um servo motor para o elevador e 3 botões que indicam os andares do prédio. Quando o botão esquerdo é pressionado o elevador deve ir para o primeiro andar que no servo motor é a posição de 0 graus, quando o botão do meio é pressionado o elevador deve ir para o segundo andar que no servo motor é a posição de 90 graus, e quando o botão da direita e pressionado o elevador deve ir para o terceiro andar que no servo motor é a posição de 180 graus.
Link: https://circuits.io/circuits/4046007-elevador

3. Sistema que identifica as cores através da reemissão da luz vinda de um objeto com um sensor LDR, também possui um Led RGB emite a cor que está sendo captada.
Link: https://circuits.io/circuits/4054351-led-rgb

4. Defina o que é teste de software e descreva o microciclo do TDD.

Teste de software é um processo, ou uma série de processos, projetados para certificar que o código do programa para o computador faz o que foi projetado para fazer (Myers et al., 2011), uma vez que o software deve ser previsível, consistente, e não apresentar comportamentos não esperados pelos usuários.

TDD se baseia em pequenos ciclos de repetições, onde para cada funcionalidade do sistema um teste é criado antes. Este novo teste criado inicialmente falha, já que ainda não temos a implementação da funcionalidade em questão e, em seguida, implementamos a funcionalidade para fazer o teste passar.
Microciclo do TDD:

1. Escrever um teste que inicialmente não passa
2. Adicionar uma nova funcionalidade do sistema
3. Fazer o teste passar
4. Refatorar o código da nova funcionalidade
5. Escrever o próximo Teste

6. Faça uma pesquisa sobre três frameworks para teste de unidade e apresente as vantagens e desvantagens de cada um, bem como, uma tabela descrevendo os itens (assertivas, análise de memory leak e outros) suportados por cada um deles.

JUnit – Java

Vantagens:

1. Permite a criação rápida de código de teste enquanto possibilita um aumento na qualidade do sistema sendo desenvolvido e testado;
2. Não é necessário escrever o próprio framework;
3. Amplamente utilizado pelos desenvolvedores da comunidade código-aberto, possuindo um grande número de exemplos;
4. Pode-se criar uma hierarquia de testes que permitirá testar apenas uma parte do sistema ou todo ele;
5. JUnit é LIVRE.

Assertions:

- AssertArrayEquals()
- AssertEquals()
- AssertFalse()
- AssertNotNull()
- AssertNotSame()
- AssertNull()
- AssertSame()
- AssertThatBothContainsString()
- AssertThatHasItems()
- AssertThatEveryItemContainsString()
- AssertThatHamcrestCoreMatchers()
- AssertTrue()

JSUnit – JavaScript

Assertions:

- assert([comment], booleanValue)
- assertTrue([comment], booleanValue)
- assertFalse([comment], booleanValue)
- assertEquals([comment], value1, value2)
- assertNotEquals([comment], value1, value2)
- assertNull([comment], value)
- assertNotNull([comment], value)
- assertUndefined([comment], value)
- assertNotUndefined([comment], value)
- assertNaN([comment], value)
- assertNotNaN([comment], value)
- fail(comment)

CppTest – C++

Assertions

- TEST_FAIL(msg)
- TEST_ASSERT(expr)
- TEST_ASSERT_MSG(expr, msg)
- TEST_ASSERT_DELTA(a, b, delta)
- TEST_ASSERT_DELTA_MSG(a, b, delta, msg)
- TEST_THROWS(expr, x)
- TEST_THROWS_MSG(expr, x, msg)
- TEST_THROWS_ANYTHING(expr)
- TEST_THROWS_ANYTHING_MSG(expr, msg)
- TEST_THROWS_NOTHING(expr)
- TEST_THROWS_NOTHING_MSG(expr, msg)