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

LISTA DE EXERCÍCIO 02

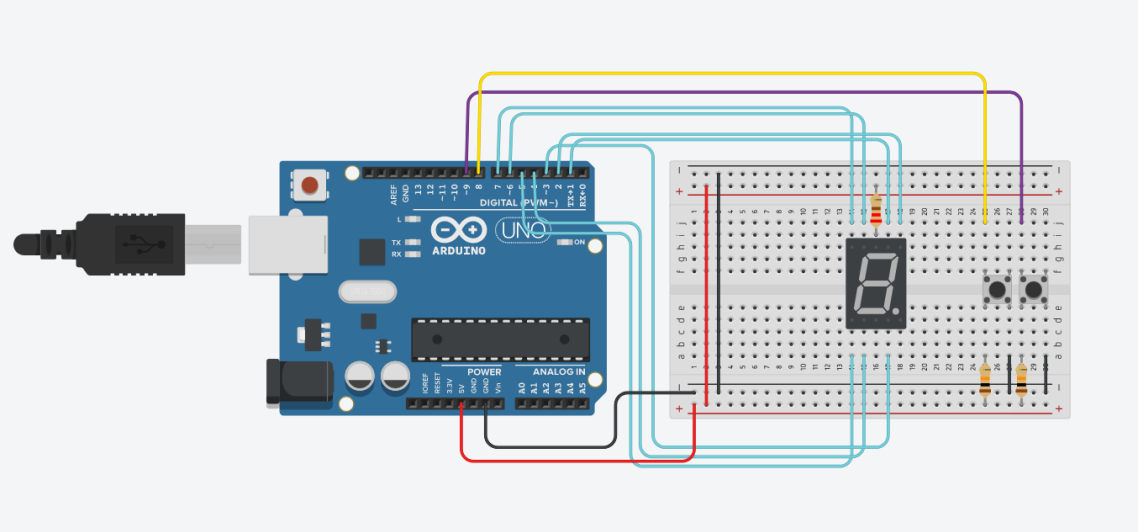
Aluno: Vinícius da Silva Dias

1. **Implemente os seguintes esquemas abaixo que deverá controlará um display de 7 segmentos que irá conectado diretamente ao Arduino e fará um contador hexadecimal configurável através de duas teclas onde você pode usá-lo de forma crescente (0-9) e decrescente (9-0).**

Descreva o resultado usando o simulador.

O Simulador utilizado foi o TinkerCad:
https://www.tinkercad.com/things/4O0dRaqRedF-questao-1-display-de-7-segmentos

Figura 01: Circuito Contador



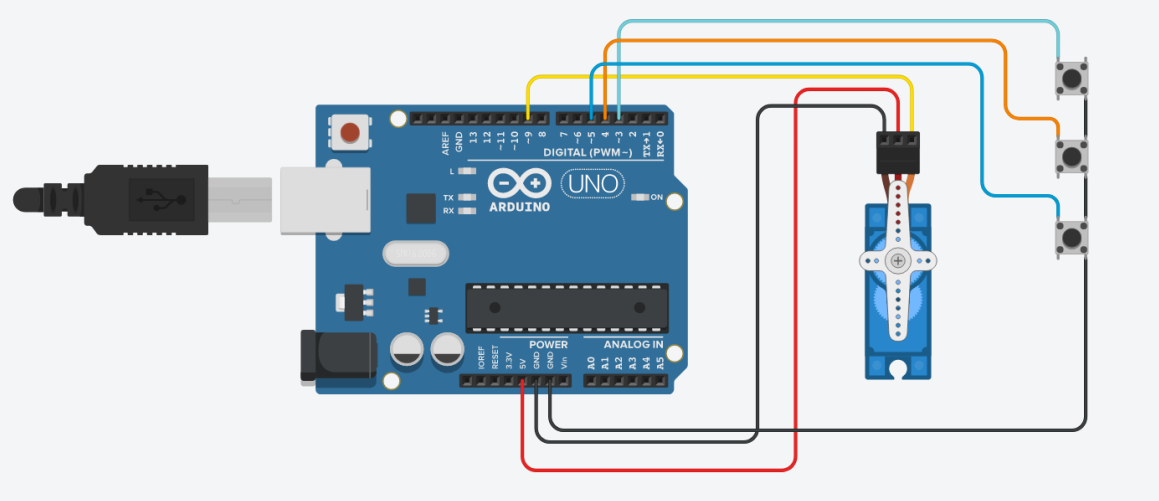
Fonte: GitHub

O código implementado consta no TinkerCad e no GitHub. A simulação apresenta o resultado do funcionamento do circuito ilustrado na figura acima. Ao pressionar o pushbutton da esquerda, o circuito funciona como um contador incrementando o valor apresentado no display de 7 segmentos. O pushbutton da direita, irá decrementar o valor apresentado no display. Estes valores seguem as variáveis do tipo “byte”, chamada tabela_7_seg, que tem a estrutura de uma matriz, onde cada linha contém o código em hexadecimal respectivo, para ligar os segmentos que vai de “a” à “g”.

2. Crie um programa para controlar um elevador que atenda 3 andares (1 botão para cada andar), onde cada andar e correspondente as seguintes posições em graus de um servo motor: andar 1 = 0º; andar 2 = 90º; e andar 3 = 180º. Apresente um esquema da ligação dos componentes necessários. Descreva o resultado usando o simulador.

O Simulador utilizado foi o TinkerCad:
https://www.tinkercad.com/things/4vxQUgX46FO-questao-2-controlador-de-um-elevador

Figura 02: Circuito controlador de Elevador



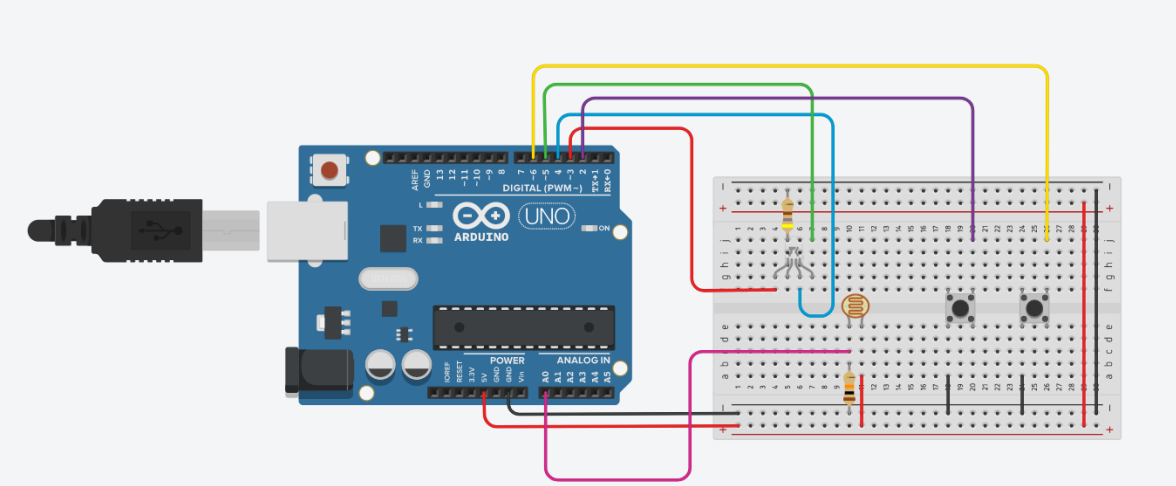
Fonte: GitHub

O código implementado consta no TinkerCad e no GitHub. A simulação descreve o funcionamento do elevador utilizando-se de três pushbuttons como controle dos andares, sendo cada um direcionado a um andar diferente, toda vez que algum deles é pressionado, o servo motor gira para o ângulo que representa o piso, térreo - 0º, piso 1 - 90º e piso 2 - 180º.

3. Implemente os seguintes esquemas abaixo que deverá identificar cores com o Arduino utilizando um LDR e um LED RGB. O programa deve gerar como saída as cores como o nome e o número em RGB. Descreva o resultado usando o simulador.

O Simulador utilizado foi o TinkerCad:
https://www.tinkercad.com/things/4E598eV3cRY-questao-3-led-rgb

Figura 03: Identificador de cores



Fonte: GitHub

O código implementado consta no TinkerCad e no GitHub. A simulação demonstrou o funcionamento de um sensor LDR como leitor de cores e o LED RGB como calibração e reprodução da cor. Inicialmente o código executa a calibração que primeiramente verifica o valor do branco e armazena em uma variável, posteriormente o valor do preto e armazena em outra variável, após esse processo o código faz a leitura do sensor LDR afim de coletar a cor posta em sua frente, posteriormente ele apresenta os valores R, G e B da cor lida e retorna ao loop de calibração, coleta e amostragem.

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

O teste de software é a etapa, na construção de um programa, responsável por checar o nível de qualidade. Os defeitos que um teste busca identificar incluem erro de compatibilidade, de algum algoritmo, de requisitos que não podem ser complementados, limitação de hardware entre outros.

Existem diferentes tipos de testes que podem ser aplicados num software para identificar suas falhas, sendo os principais:

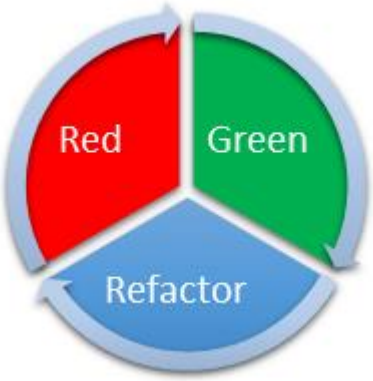
- – Teste da caixa branca;
- – Teste da caixa preta;
- – Teste da caixa cinza;
- – Teste de regressão;
- – Teste de unidade;
- – Teste de integração;
- – Teste de carga;
- – Teste de usabilidade;
- – Teste de stress;

Para que esses testes possam ser realizados de modo mais rápido e com maior abrangência, existem ferramentas que automatizam alguns deles ou auxiliam na execução de outros. Essas são as ferramentas de teste de software.

O Test Driven Development é uma metodologia de desenvolvimento avançado com foco na qualidade do software orientado a objetos, amplamente utilizado dentro da comunidade Agile. O que torna o TDD único é o seu foco na especificação do projeto, ao invés de ser na sua validação.

A abordagem TDD leva a uma melhor arquitetura de software de qualidade, melhorando a qualidade, simplicidade e capacidade de teste, enquanto aumenta a confiança no código e a produtividade da equipe. Ao implementarmos essa metodologia, os testes utilizados são compostos de afirmações verdadeiras e falsas que indicam o comportamento correto com relação ao teste que está sendo usado no momento.

Figura 04 – Cliclo TDD.



Fonte: https://www.devmedia.com.br/introducao-ao-desenvolvimento-guiado-por-testes/33112

O TDD é conduzido por uma filosofia básica, conhecida como "Red-Green-Refactor" (Figura 5), que é um microciclo de desenvolvimento iterativo no qual os testes e comportamentos são implementados.

Neste processo, o que ocorre primeiramente é que os casos de teste recém-escritos testam o código ainda não escrito e, devido a isso, falham. Esta é a primeira etapa, definida como Red.

A partir deste momento o desenvolvedor implementa o recurso da maneira mais simples, fazendo com que os testes correspondentes sejam bem-sucedidos, o que representa o segundo passo do ciclo, o Green.

E por fim, antes de começarmos com um novo teste, podemos realizar o processo de refatoração do código de forma a deixá-lo o mais claro e eficiente possível, de forma a proporcionar confiança pela capacidade de executar o teste novamente que foi implementado.

Este é o terceiro item do ciclo, o Refactor. Ao adotarmos o ciclo Red-Green-Refactor, temos que o nosso código pode evoluir para um nível superior de comunicação simplificada, confiança e produtividade global.