Relatório do Trabalho Prático Individual

Introdução

Foi-nos solicitado, no âmbito da cadeira de Tecnologia da Informática, o desenvolvimento do jogo Super Bit Smasher como parte de um trabalho prático. O objetivo principal do jogo é transformar um valor inicial em um valor-alvo usando operações lógicas bitwise (AND, OR, XOR). O jogador interage com o sistema através de botões físicos para selecionar a operação desejada, enquanto LEDs indicam o tempo restante para concluir a rodada. O programa implementa um controle preciso de entrada para evitar leituras inconsistentes (debouncing) e inclui uma funcionalidade de reinício ativada por um press prolongado de um botão. O jogo foi projetado para rodadas sucessivas, apresentando novos desafios por meio de valores gerados aleatoriamente e limites de tempo controlados.

1. Estrutura e Inicialização

A estrutura do código foi cuidadosamente planejada para organizar as principais funcionalidades do jogo. Variáveis foram definidas para gerenciar os botões, LEDs e o tempo limite. Os botões são configurados nos pinos digitais 2, 3 e 4 como INPUT_PULLUP, enquanto os LEDs estão conectados aos pinos 8 a 11. Essa configuração permite que o sistema reconheça facilmente as interações físicas dos jogadores. No método “setup()”, os LEDs são inicializados como ligados, sinalizando que o jogo está pronto para começar. A função “resetGame()” é chamada no início para garantir que o sistema esteja preparado para a primeira rodada.

2. Função Principal (Loop)

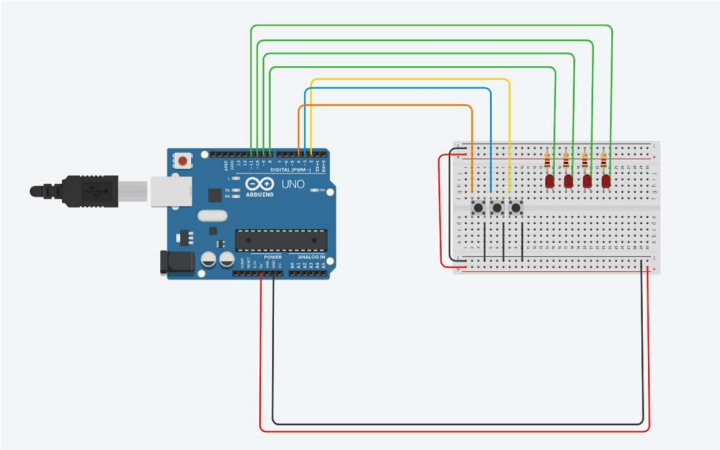
A função principal, implementada no método “loop()”, gerencia o ciclo contínuo do jogo. Inicialmente, verifica se o botão OR foi pressionado por 2 segundos, usando a função “checkLongPress()”. Caso essa condição seja atendida, o jogo é reiniciado. Além disso, o código monitora a entrada serial para capturar números inseridos pelo jogador, que serão usados nas operações lógicas. Quando um botão correspondente a uma operação lógica é pressionado, a função “applyOperation()” é acionada para realizar a transformação do valor inicial com base na operação escolhida.

3. Funções do Jogo

As funcionalidades principais do jogo foram implementadas de forma modular para melhorar a organização do código e facilitar a manutenção. A função “resetGame()” é responsável por reiniciar os valores do jogo, determinar a disponibilidade das operações lógicas (AND e XOR), e acender todos os LEDs. A função “updateLeds()” foi implementada para gerenciar os LEDs, que indicam visualmente o tempo restante, apagando-os progressivamente em intervalos iguais. A lógica por trás das operações bitwise foi organizada na função “applyOperation()”, que processa o valor atual com o número inserido pelo jogador usando a operação selecionada. O controle de debounce, essencial para evitar leituras erradas dos botões, foi realizado com a função “debouncedRead()”.

4. Lógica de Vitória

A lógica de vitória do jogo verifica se o valor inicial foi transformado no valor-alvo após as operações lógicas. Quando essa condição é atingida, uma mensagem de sucesso é exibida ao jogador, e o jogo reinicia automaticamente com novos valores aleatórios. Isso garante uma experiência contínua e desafiadora para o jogador.



Circuito tinkercad

Conclusão

Este trabalho resultou na criação de um jogo interativo e funcional, atendendo a todos os requisitos estabelecidos. O uso de botões físicos e LEDs para representar as interações do jogo proporciona uma experiência imersiva para o jogador. As funcionalidades foram implementadas de forma modular, priorizando a clareza do código e a eficiência. A lógica de debounce e o gerenciamento de tempo asseguram que o jogo opere sem falhas, mesmo em situações de interação rápida. O projeto demonstrou a importância de planejar bem a estrutura do código para integrar diferentes componentes de hardware e software.