MÁQUINA DE TURING - GRUPO 3

Murilo Fontana Muniz[*], Vinicius Lopes Silvino[†], Rafael Ribas de Lima[‡]

Emails: Murilo.2022@alunos.utfpr.edu.br, vsilvino@alunos.utfpr.edu.br, rafael.2022@alunos.utfpr.edu.br

Resumo— O presente trabalho tem como objetivo principal demonstrar a aplicação de uma Máquina de Turing para processar a linguagem $L = 14^*1, \Sigma = \{1, 4, 5\}$ finalizando com o símbolo 5. Utilizando as ferramentas JFLAP e Tinkercad, foram configurados os estados e transições necessários para que a Máquina de Turing processasse a sequência especificada. A metodologia envolveu a montagem da Máquina de Turing para simulação, seguida pela programação com a lógica necessária para a linguagem alvo e testes práticos para validar o seu funcionamento, realizando ajustes conforme necessário. Como resultado, foi possível ilustrar claramente como uma Máquina de Turing pode reconhecer padrões específicos, reforçando a compreensão teórica e prática dos conceitos de computabilidade.

Palavras-chave— Turing, linguagem

1 Introdução

A teoria da computação é um campo fundamental da ciência da computação que explora as capacidades e limitações dos modelos de computação. Entre esses modelos, a Máquina de Turing, proposta por Alan Turing em 1936, ocupa um lugar central devido à sua simplicidade e poder expressivo. Segundo Sipser (2006), a Máquina de Turing é um modelo abstrato que utiliza uma fita infinita e um conjunto de regras para manipular símbolos, sendo capaz de simular qualquer algoritmo computacional. De acordo com ele, "a máquina de Turing é um modelo de um computador, mas sua definição é precisa e se presta bem a estudos formais". Este conceito não apenas oferece uma compreensão teórica profunda, mas também estabelece as bases para o desenvolvimento de linguagens de programação e sistemas computacionais modernos.

Este trabalho tem como objetivo principal apresentar uma simulação de uma Máquina de Turing que processa a linguagem específica definida por

$$L = 14^*1, \sum = \{1, 4\}$$

e finalizada com o símbolo 5. Esta simulação foi implementada utilizando o JFLAP, uma ferramenta educacional utilizada para ensinar conceitos de autômatos e teoria da computação. A escolha dessa linguagem permite demonstrar a capacidade da Máquina de Turing em reconhecer e processar padrões complexos, ilustrando a aplicação prática de conceitos teóricos fundamentais.

A metodologia empregada envolveu a definição de estados e transições que permitem à máquina processar a sequência de entrada de forma correta. Sipser afirma que "uma máquina de Turing é definida por uma lista de estados, uma fita dividida em células e um conjunto de instruções de transição". A fita da máquina foi configurada para aceitar entradas compostas por uma série de '1's seguidos por '4's, um '1' final e terminando com o símbolo '5'. Após a configuração, a máquina foi submetida a uma série de testes para validar seu funcionamento, com ajustes finos sendo realizados conforme necessário para garantir precisão e conformidade com a linguagem alvo.

A simulação realizada não só reforça a compreensão teórica da Máquina de Turing, mas também oferece uma ferramenta prática para estudantes e profissionais. Segundo Sipser, "estudos de caso de processos específicos de máquinas de Turing são úteis na compreensão de conceitos teóricos". Ao demonstrar como a Máquina de Turing pode ser utilizada para resolver problemas específicos de reconhecimento de padrões, este trabalho contribui para a educação e a aplicação prática dos princípios fundamentais da teoria da computação.

2 Métodologia

A metodologia de implementação envolveu três etapas principais: (1) o design de um diagrama de estados projetado para a linguagem no JFLAP, (2) a construção de um protótipo físico no Tinkercad, incluindo a integração de um teclado matricial para entrada de caracteres e um display LCD I2C para exibir o status e operação da máquina, e (3) a simulação e teste da máquina para garantir a precisão no reconhecimento na linguagem '14*1'.

1. Para fazer o diagrama de estados no JFLAP, como mostrado no vídeo da Neso Academy, e com adaptações para o modelo conforme necessário para a linguagem. O tutorial facilitou a configuração inicial dos estados e transições.

   No JFLAP, foram criados os estados e transições essenciais para processar a sequência de entrada. O quadro de estados foi de extrema importância, oferecendo uma representação visual clara das transições, o que facilitou a visualização das transições, a compreensão e a depuração do código do autômato.

2. Para a construção do protótipo foi utilizada uma lógica de leitura de um 'teclado matri-

cial 4x4' (Figura X), integrado à um Arduino UNO e este projetando os caracteres necessários no LCD I2C. Se fazendo vantagem da lógica do teclado (detalhada em DESENVOLVIMENTO), foi-se desenvolvido um código em C++, como detalhado no Apêndice A (Silvino; Muniz; Ribas, 2024), para conseguir interpretar as entradas da máquina simulada pelo protótipo. Ademais, no mesmo código foram adicionadas as exibições das entradas por meio de um LCD com um decodificador I2C nativo.

3. Para validar o funcionamento da Máquina de Turing, utilizamos a plataforma Tinkercad para simulações práticas. Primeiramente, iniciamos a simulação no Tinkercad, onde o código previamente configurado começa a ser executado. Uma vez que o LCD esteja ligado e pronto para uso, digitamos a palavra desejada no teclado matricial.

   O teclado matricial permite a entrada de números entre 0 e 9, letras de A à D e os caracteres " * " e " # ". O número '5' funciona como a tecla "Enter", indicando à Máquina de Turing que a sequência de entrada está completa e pronta para ser processada. Ao digitar '5', a máquina lê a entrada e processa a palavra de acordo com os estados e transições configurados.

   O LCD exibirá "Accepted!"se a palavra estiver de acordo com a linguagem, ou "Rejected!"se a palavra não atender aos critérios definidos. Esse feedback imediato permite verificar a correta implementação e funcionamento. Foi testado várias sequências de entrada, incluindo diferentes combinações de números. Isso garantiu uma validação abrangente do comportamento da máquina para múltiplos casos possíveis.

   Após 3 segundos de cada teste, a máquina fica pronta para receber uma nova entrada, permitindo uma rápida interação. Esta abordagem prática facilitou a identificação e correção de qualquer inconsistência na configuração dos estados e transições, assegurando a precisão e confiabilidade da Máquina de Turing simulada.

Tendo o diagrama JFLAP, foram criadas várias palavras teste para serem comparados os resultados entre a saída do JFLAP e as saídas do protótipo.

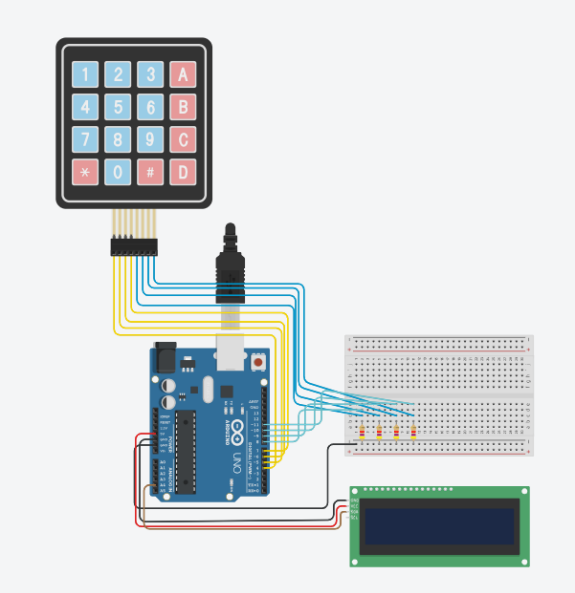
3 Desenvolvimento

A implementação da teoria da Máquina de Turing deu-se no Tinkercad, e foi feito uso dos seguintes materiais:

- **Arduino Uno R3**: Microcontrolador utilizado para controlar todo o sistema.
- **Teclado Matricial 4x4**: Teclado de 16 teclas (0-9, A-D, *, #) para entrada de dados.
- **Display LCD 16x2**: Tela de cristal líquido para exibir informações, geralmente de 16 colunas e 2 linhas.
- **Breadboard**: Placa de ensaio para facilitar a montagem do circuito sem a necessidade de solda.
- **Resistores**: Resistores de 4.70 KΩ, utilizados para limitar corrente ou configurar o teclado e display LCD.
- **Fios de Conexão (Jumpers)**: 12 fios para realizar as conexões elétricas entre os componentes e o Arduino.

A seguir, é mostrado o projeto da Máquina de Turing.

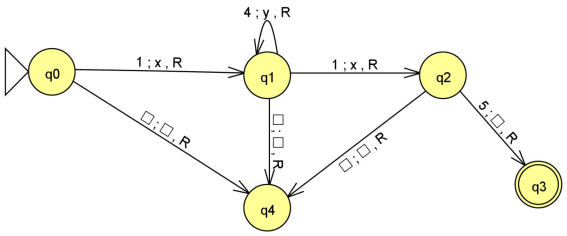
Figura 1: Maquina de Turing projeto Tinkercad.



Fonte: Autoria própria

Para desenvolver a lógica para máquina de Turing, foi projetado um diagrama, explicitando as mudanças de estados e suas necessidades.

Figura 3: Diagrama de Estados desenvolvido no JFLAP.



Fonte: Autoria própria

Na sequência discute-se linhas de código utilizadas para implementar na arquitetura desenvolvida, conforme o Apêndice A.

As 20 primeiras linhas reference respectivamente às bibliotecas as quais são usadas para comunicação I2C e controle do display LCD, inicializador do LCD com endereço I2C 0x27 e tamanho de 16 colunas e 2 linhas, definição dos pinos para linhas e colunas do teclado e algumas variáveis auxiliares, enumeração dos possíveis estados da máquina de Turing e inicializa o estado atual como q0 e string usada como fita para a máquina de Turing. Segue uma descrição das funções contidas no código:

Função setup: Configura o LCD, inicializa a comunicação serial e define os modos dos pinos para o teclado matricial.

Função loop: Lê entradas do teclado, verificando cada linha e coluna, e chama processInput se uma tecla for pressionada.

Função processInput: Processa a entrada do teclado. Se a tecla '5' for pressionada, inicia a máquina de Turing; caso contrário, adiciona o caractere à fita e o exibe no LCD.

Função getInputChar: Retorna o caractere correspondente à combinação de linha e coluna do teclado.

Função runTuringMachine: Executa a máquina de Turing com base na fita de entrada. A máquina de Turing move-se entre estados q0, q1, q2, q3, $q_accepteq_reject$, modificando a fita conforme necessário.Exibe no LCD se a entrada foi aceita ou rejeitada.

Função displaywrite: Exibe caracteres no LCD. Se o número de caracteres exceder o limite do LCD, limpa a tela e começa a exibir novamente a partir do início.

Para os testes de prova, foram escolhidas as palavras: '141', '1441', '441', '1481', '14', '1' e '4'. Colocando os resultados obtidos para comparação com os gerados utilizando o JFLAP, obtiveram-se o seguintes resultados:

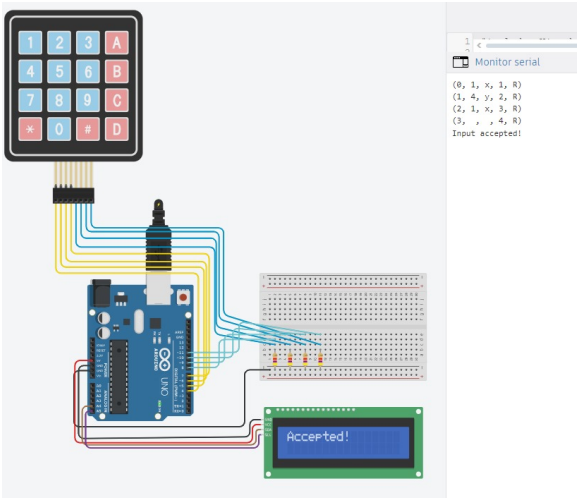


Figura 5: Resultado da palavra '141' no monitor serial.

Fonte: Autoria própria.

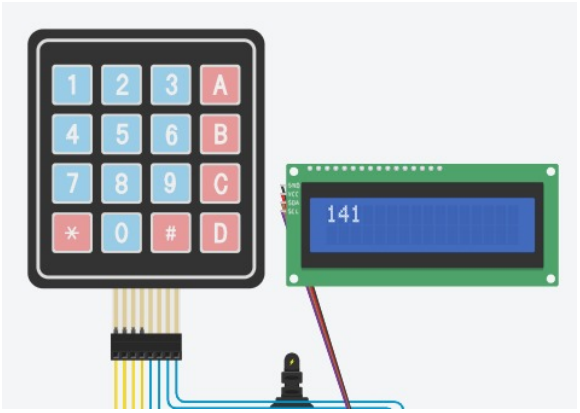


Figura 6: Exibição da palavra '141' no LCD.

Fonte: Autoria própria.

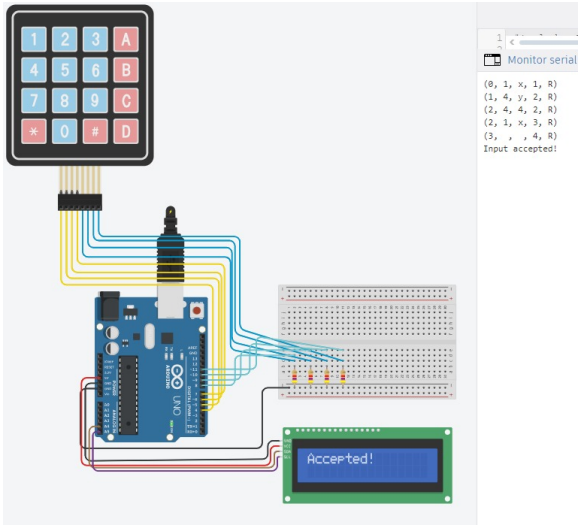


Figura 7: Resultado da palavra ’1441’ no monitor serial.

Fonte: Autoria própria.

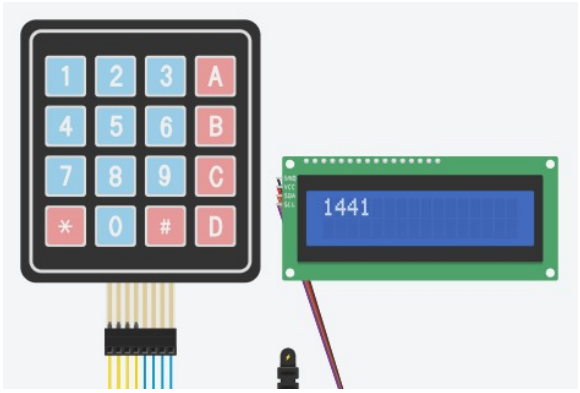


Figura 8: Exibição da palavra ’1441’ no LCD.

Fonte: Autoria própria.

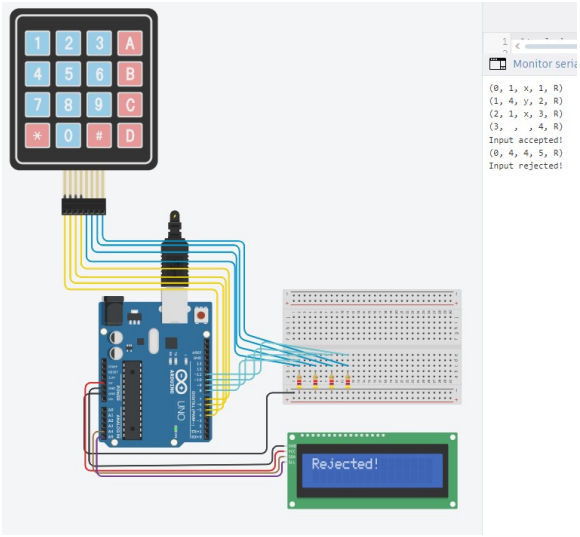


Figura 9: Resultado da palavra ’441’ no monitor serial.

Fonte: Autoria própria.

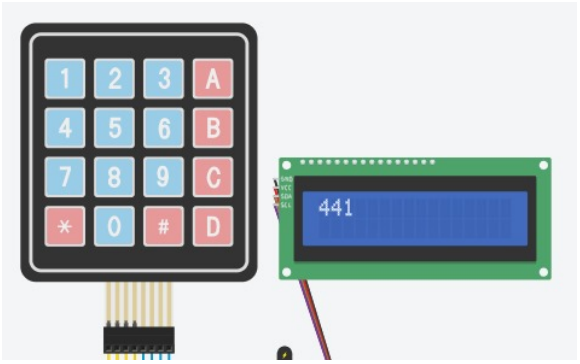


Figura 10: Exibição da palavra ’441’ no LCD.

Fonte: Autoria própria.

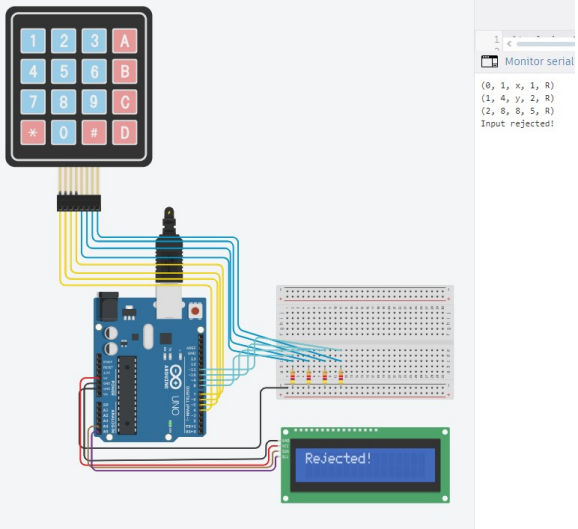


Figura 11: Resultado da palavra ’1481’ no monitor serial.

Fonte: Autoria própria.

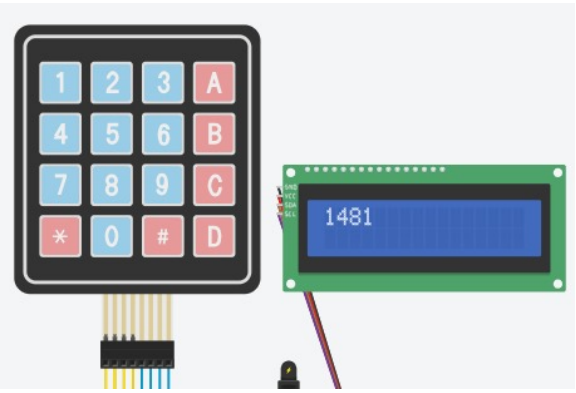


Figura 12: Exibição da palavra ’1481’ no LCD.

Fonte: Autoria própria.

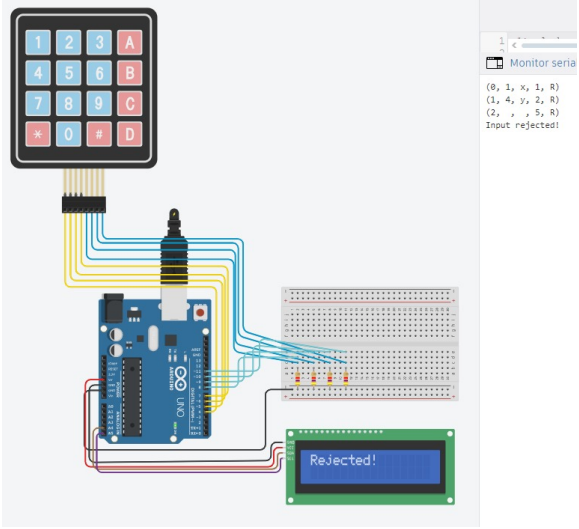


Figura 13: Resultado da palavra ’14’ no monitor serial.

Fonte: Autoria própria.

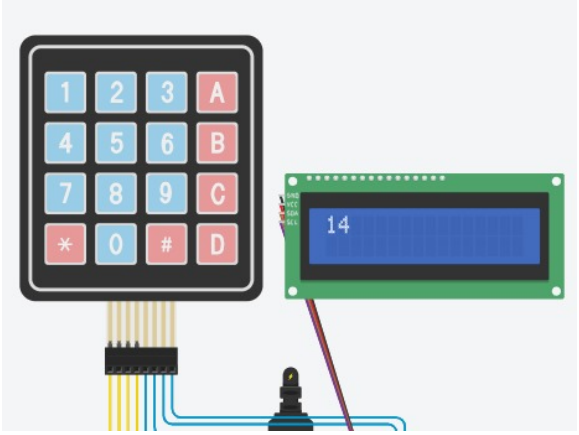


Figura 14: Exibição da palavra ’14’ no LCD.

Fonte: Autoria própria.

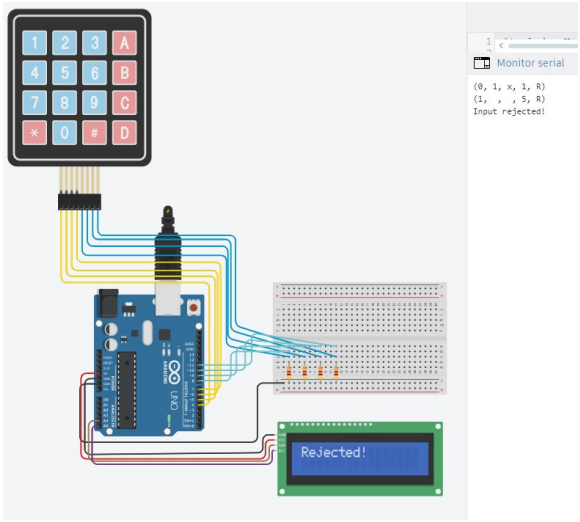


Figura 15: Resultado da palavra '1' no monitor serial.

Fonte: Autoria própria.

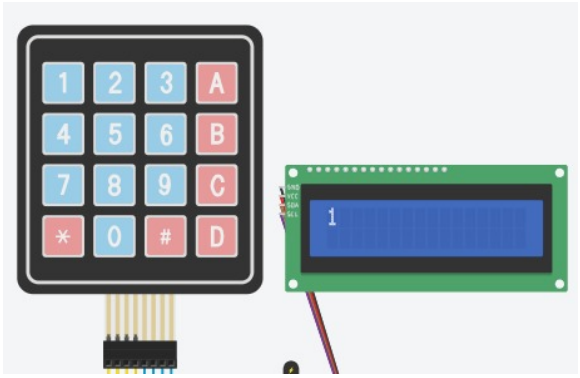


Figura 16: Exibição da palavra '1' no LCD.

Fonte: Autoria própria.

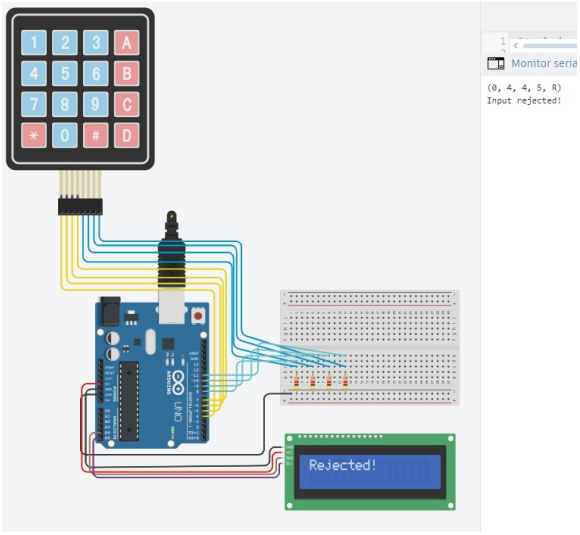


Figura 17: Resultado da palavra '4' no monitor serial.

Fonte: Autoria própria.

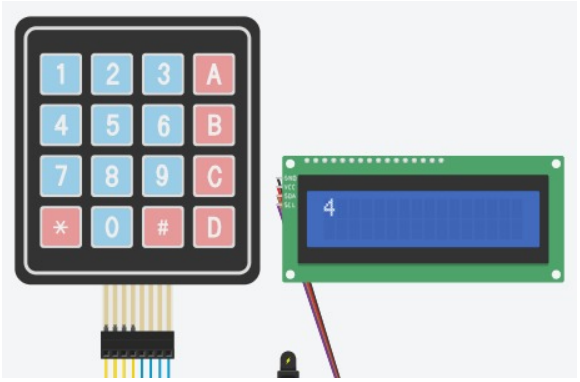


Figura 18: Exibição da palavra '4' no LCD.

Fonte: Autoria própria.

4 Considerções

Este trabalho demonstrou a aplicação de uma Máquina de Turing para processar a linguagem

$L = 14^*1, \sum = 1, 4$

finalizando com o símbolo 5. Utilizando JFLAP e Tinkercad, configuramos e testamos a máquina, mostrando a eficácia dessas ferramentas na simulação de conceitos teóricos de computação. A montagem e programação da Máquina de Turing, aliadas aos testes práticos, validaram seu funcionamento correto e ressaltaram seu papel fundamental na teoria da computação. A experiência prática proporcionou uma compreensão aprofundada dos mecanismos de reconhecimento de padrões e demonstrou a utilidade das Máquinas de Turing na resolução de problemas específicos. Concluímos que a simulação de Máquinas de Turing é uma abordagem valiosa para o ensino e

a aprendizagem de conceitos de computação, contribuindo para o desenvolvimento de habilidades práticas em programação e análise de autômatos.

Referências

C., G. **Circuit design LCD 16x2 (I2C) - Tinkercad**. Disponível em: <https://www.tinkercad.com/things/1qtYYihv9B5-lcd-16x2-i2c>. Acesso em: 8 jun. 2024.

NESO ACADEMY. **Turing Machine (Example 1)**. YouTube, 12 set. 2017. Disponível em: <https://www.youtube.com/watch?v=D9eF_B8URnw>. Acesso em: 8 jun. 2024.

GE. **Como Ligar Teclado Matricial 4x4 no Display LCD 16x2 e no Monitor Serial - Tutorial 7**. YouTube, 2 nov. 2020. Disponível em: <https://www.youtube.com/watch?v=bm_3IVHHHiE>. Acesso em: 8 jun. 2024.

Sipser, M. (2006). **Introdução à Teoria da Computação**. São Paulo, SP: Thomson Learning.

Apêndices

Apêndice A: Código em C++ do protótipo

Código em C++ utilizado para interpretar as entradas da máquina simulada pelo protótipo.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Iniciar display no endereço 0x27
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Definir linhas e colunas
int l1 = 4, l2 = 5, l3 = 6, l4 = 7;
int c1 = 8, c2 = 9, c3 = 10, c4 = 11;
bool final_line = false;

// var para \n no lcd:
int _n = 0;

// Definir estados da máquina de Turing
enum State { q0, q1, q2, q3, q_accept, q_reject, q_end };
State currentState = q0;

// Definir fita
String tape = "";

void setup() {
  lcd.init();
  lcd.backlight();
  delay(2000);
  lcd.clear();
  lcd.home();
  Serial.begin(9600);

  // Definir todas as linhas como saída
  pinMode(l1, OUTPUT);
  pinMode(l2, OUTPUT);
  pinMode(l3, OUTPUT);
  pinMode(l4, OUTPUT);

  // Definir todas as colunas como entrada
  pinMode(c1, INPUT);
  pinMode(c2, INPUT);
  pinMode(c3, INPUT);
  pinMode(c4, INPUT);
}

void loop() {
  for (int lX = 4; lX < 8; lX++) {
    // Definir todas as linhas do teclado como LOW
    digitalWrite(l1, LOW);
    digitalWrite(l2, LOW);
    digitalWrite(l3, LOW);
    digitalWrite(l4, LOW);

    // Definir linha X como HIGH
    digitalWrite(lX, HIGH);

    // Ler colunas
    if (digitalRead(c1) == HIGH) {
      processInput(lX, c1);
      while (digitalRead(c1) == HIGH) {}
    }
    if (digitalRead(c2) == HIGH) {
      processInput(lX, c2);
      while (digitalRead(c2) == HIGH) {}
    }
    if (digitalRead(c3) == HIGH) {
      processInput(lX, c3);
      while (digitalRead(c3) == HIGH) {}
    }
    if (digitalRead(c4) == HIGH) {
      processInput(lX, c4);
      while (digitalRead(c4) == HIGH) {}
    }
  }
}

void processInput(int line, int column) {
  char input = getInputChar(line, column);
  if (input == '5') {
    runTuringMachine();
  } else {
    tape += input;
    display_write(input);
  }
}

char getInputChar(int line, int column) {
  if (line == 4 && column == 8) return '1';
  if (line == 4 && column == 9) return '2';
  if (line == 4 && column == 10) return '3';
  if (line == 4 && column == 11) return 'A';
  if (line == 5 && column == 8) return '4';
  if (line == 5 && column == 9) return '5'; // Finalizador
  if (line == 5 && column == 10) return '6';
  if (line == 5 && column == 11) return 'B';
  if (line == 6 && column == 8) return '7';
  if (line == 6 && column == 9) return '8';
  if (line == 6 && column == 10) return '9';
  if (line == 6 && column == 11) return 'C';
  if (line == 7 && column == 8) return '*';
  if (line == 7 && column == 9) return '0';
  if (line == 7 && column == 10) return '#';
  if (line == 7 && column == 11) return 'D';
  return ' ';
}

void runTuringMachine() {
  int tapeIndex = 0;
  currentState = q0;

  while (currentState != q_accept && currentState !=
  q_reject) {
    char currentSymbol = (tapeIndex < tape.length()) ?
    tape[tapeIndex] : ' ';
    char writeSymbol = currentSymbol;
    State nextState = currentState;
    String direction = "R";
```

```
  switch (currentState) {
    case q0:
      if (currentSymbol == '1') {
        nextState = q1;
        writeSymbol = 'x';
      } else {
        nextState = q_reject;
      }
      break;

    case q1:
      if (currentSymbol == '4') {
        nextState = q2;
        writeSymbol = 'y';
      } else {
        nextState = q_reject;
      }
      break;

    case q2:
      if (currentSymbol == '4') {
        nextState = q2;
      } else if (currentSymbol == '1') {
        nextState = q3;
        writeSymbol = 'x';
      } else {
        nextState = q_reject;
      }
      break;

    case q3:
      if (currentSymbol == ' ' || tapeIndex == tape.length()) {
        nextState = q_accept;
      } else {
        nextState = q_reject;
      }
      break;

    default:
      nextState = q_reject;
      break;
  }

  // Exibir quíntupla no monitor serial
  Serial.print("(");
  Serial.print(currentState);
  Serial.print(", ");
  Serial.print(currentSymbol);
  Serial.print(", ");
  Serial.print(writeSymbol);
  Serial.print(", ");
  Serial.print(nextState);
  Serial.print(", ");
  Serial.print(direction);
  Serial.println(")");

  if (nextState == q_reject) {
    currentState = q_reject;
    break;
  }

  // Atualizar estado e mover a cabeça de leitura
  if (tapeIndex < tape.length()) {
    tape[tapeIndex] = writeSymbol;
  } else {
    tape += writeSymbol;
  }
  currentState = nextState;
  tapeIndex += (direction == "R" ? 1 : -1);
}

if (currentState == q_accept) {
  Serial.println("Input accepted!");
  lcd.clear();
  lcd.print("Accepted!");
} else {
  Serial.println("Input rejected!");
  lcd.clear();
  lcd.print("Rejected!");
}

// Esperar 3 segundos antes de limpar a fita e o LCD
delay(3000);
tape = "";
lcd.clear();
lcd.home();
_n = 0;
final_line = false;

currentState = q_end;
}

void display_write(char input) {
  if (_n == 16) {
    lcd.setCursor(0, 1);
  }

  if (_n == 32) {
    final_line = true;
    _n = 1;
  }

  if (final_line == true) {
    lcd.clear();
    lcd.home();
    lcd.print(".");
    final_line = false;
  }

  lcd.print(input);
  _n++;
}
```