Gustavo dos Santos Nunes

Implementação de uma Máquina de Turing utilizando Arduíno

Apucarana - Pr
14 de fevereiro de 2025

Sumário

0.1 Introdução

Este relatório apresenta o desenvolvimento de uma Máquina de Turing implementada utilizando a plataforma Arduino. O sistema permite a entrada de regras de transição via monitor serial, e de palavras via teclado, proporcionando uma abordagem interativa para simulação de uma Máquina de Turing. A relevância deste projeto reside na aplicação prática dos conceitos teóricos da computação.

0.2 Fundamentação Teórica

0.2.1 Máquina de Turing

A Máquina de Turing é um modelo teórico de computação que consiste em uma fita infinita e um cabeçote que pode ler, escrever e mover-se pela fita de acordo com um conjunto de regras de produção. As saídas de aceite e rejeite são apresentadas de acordo com as regras nela configurada, caso nenhuma das duas saídas forem satisfeitas ela continuara para sempre nunca parando.

0.2.2 Arduino

O Arduino é uma plataforma de prototipagem eletrônica de código aberto que facilita o desenvolvimento de projetos interativos. Neste projeto, o Arduino é utilizado para implementar a lógica da Máquina de Turing e a interface de entrada via teclado.

0.3 Metodologia

0.3.1 Hardware Utilizado

- **Arduino Uno**: Microcontrolador principal.
- **Teclado Matricial**: Para entrada das regras de produção.
- **Display LCD 16x2**: Para visualização do estado atual da fita e outras informações.
- **Protoboard**: Usado para melhor organização dos componentes.
- **Potenciometro 10k**
- **Resistor 220Ω**
- **Jumpers**: Usados para conectar os componentes.

0.3.2 Estrutura do Código

O código foi desenvolvido de forma modular, permitindo a leitura das regras de produção via serial monitor e a execução do algoritmo que simula a Máquina de Turing. A seguir, apresenta-se agora o codigo completo da maquina de Turing:

```
#include <LiquidCrystal.h>
#include <Keypad.h>

// Configura  o do LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Configura  o Fixa do Teclado Matricial
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  { '0', '1', ' ', 'a' },
  { '*', '#', ' ', 'b' },
  { '&', '$', ' ', 'c' },
  { ' ', ' ', ' ', 'd' }
};
byte rowPins[ROWS] = {14, 15, 16, 17};
byte colPins[COLS] = {6, 7, 8, 9};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Estrutura para regras de transi  o
struct TransitionRule {
  char currentState;
  char readSymbol;
  char nextState;
  char writeSymbol;
  char direction; // 'E' para esquerda, 'D' para direita, 'S' para parar
};

// Vari veis globais
TransitionRule rules[20]; // Array para armazenar at  20 regras
int ruleCount = 0;        // Contador de regras
char initialState = '\0'; // Estado inicial (inicializado com valor nulo
    )
char finalState = '\0';   // Estado final (inicializado com valor nulo)
String tape = "";         // Fita
int headPosition = 0;     // Posi  o da cabe a
char currentState = '\0'; // Estado atual (inicializado com valor nulo)

// Fun  o para exibir a fita e o estado atual no LCD
void displayTape() {
  lcd.clear();
  lcd.setCursor(0, 0);
```

```
42    lcd.print("Tape: ");
43
44    for (int i = 0; i < tape.length(); i++) {
45      if (i == headPosition) {
46        lcd.print("[");
47        lcd.print(tape[i]);
48        lcd.print("]");
49      } else {
50        lcd.print(tape[i]);
51      }
52    }
53
54    lcd.setCursor(0, 1);
55    lcd.print("State: ");
56    if (currentState != '\0') { // Verifica se o estado atual    v lido
57      lcd.print(currentState);
58    } else {
59      lcd.print("N/A"); // Exibe "N/A" se o estado for inv lido
60    }
61  }
62
63  // Fun   o para processar a fita com base nas regras de transi   o
64  void processTape() {
65    if (initialState == '\0' || finalState == '\0') {
66      lcd.clear();
67      lcd.print("Estados nao config!");
68      return;
69    }
70
71    currentState = initialState;
72    headPosition = 0;
73
74    while (true) {
75      displayTape();
76      delay(500);
77
78      char currentSymbol = tape[headPosition];
79      bool ruleFound = false;
80
81      for (int i = 0; i < ruleCount; i++) {
82        if (rules[i].currentState == currentState && rules[i].readSymbol
             == currentSymbol) {
83          currentState = rules[i].nextState;
84          tape[headPosition] = rules[i].writeSymbol;
85
86          // Movimenta   o da cabe a
87          if (rules[i].direction == 'E') {
```

```
88            headPosition = (headPosition > 0) ? headPosition - 1 : 0;
89          } else if (rules[i].direction == 'D') {
90            headPosition = (headPosition < tape.length() - 1) ?
                 headPosition + 1 : tape.length() - 1;
91          }
92
93          // Verifica se atingiu o estado final (independente da dire  o
               )
94          if (currentState == finalState) {
95            displayTape();
96            lcd.setCursor(0, 1);
97            lcd.print("ACEITA!");
98            return;
99          }
100
101         ruleFound = true;
102         break;
103       }
104     }
105
106     if (!ruleFound) {
107       displayTape();
108       lcd.setCursor(0, 1);
109       lcd.print("REJEITADA!");
110       return;
111     }
112   }
113 }
114
115 // Fun  o para adicionar uma nova regra de transi  o
116 void addRule(char currentState, char readSymbol, char nextState, char
       writeSymbol, char direction) {
117   if (ruleCount < 20) {
118     rules[ruleCount] = {currentState, readSymbol, nextState, writeSymbol
          , direction};
119     ruleCount++;
120   }
121 }
122
123 // Fun  o para processar todas as regras recebidas pelo Serial
124 void processRules(String content) {
125   lcd.clear();
126   lcd.print("Processando regras...");
127
128   // Reinicializa as vari veis
129   ruleCount = 0;
130   initialState = '\0';
```

```
131   finalState = '\0';
132
133   // Divide o conte do em partes separadas por ';'
134   int start = 0;
135   int end = content.indexOf(';');
136   while (end != -1 && ruleCount < 20) {
137     String line = content.substring(start, end);
138     line.trim(); // Remove espa os em branco e caracteres especiais
139
140     // Ignora linhas vazias
141     if (line.length() == 0) {
142       start = end + 1;
143       end = content.indexOf(';', start);
144       continue;
145     }
146
147     // Processa o estado inicial
148     if (line.startsWith("initialState:")) {
149       initialState = line.charAt(line.indexOf(':') + 1); // Pega o
              caractere ap s ':'
150       lcd.clear();
151       lcd.print("Initial: ");
152       lcd.print(initialState);
153     }
154     // Processa o estado final
155     else if (line.startsWith("finalState:")) {
156       finalState = line.charAt(line.indexOf(':') + 1); // Pega o
              caractere ap s ':'
157       lcd.clear();
158       lcd.print("Final: ");
159       lcd.print(finalState);
160     }
161     // Processa as regras de transi  o
162     else if (line.indexOf(',') != -1) {
163       // Formato da regra: q0,0,q1,1,R
164       char currentState = line.charAt(0);
165       char readSymbol = line.charAt(2);
166       char nextState = line.charAt(4);
167       char writeSymbol = line.charAt(6);
168       char direction = line.charAt(8);
169       addRule(currentState, readSymbol, nextState, writeSymbol,
              direction);
170     }
171
172     start = end + 1;
173     end = content.indexOf(';', start);
174   }
```

```
175
176   lcd.clear();
177   if (initialState != '\0' && finalState != '\0') {
178     lcd.print("Regras carregadas!");
179   } else {
180     lcd.print("Erro nas regras!");
181   }
182 }
183
184 void setup() {
185   // Inicializa o LCD
186   lcd.begin(16, 2);
187   lcd.print("Aguardando regras...");
188
189   // Inicializa o Serial
190   Serial.begin(115200);
191   while (!Serial) {
192     ; // Aguarda a conex o Serial
193   }
194 }
195
196 void loop() {
197   // Verifica se h  dados dispon veis no Serial
198   if (Serial.available()) {
199     String content = Serial.readStringUntil('\n'); // L  toda a entrada
          at  a quebra de linha
200     content.trim(); // Remove espa os em branco e caracteres especiais
201     processRules(content); // Processa todas as regras
202   }
203
204   // Verifica as entradas do teclado matricial
205   char key = keypad.getKey();
206   if (key) {
207     if (key == '#') { // Tecla '#' para processar a fita
208       processTape();
209     } else if (key == '*') { // Tecla '*' para limpar a fita
210       tape = "";
211       headPosition = 0;
212       currentState = initialState;
213       displayTape();
214     } else {
215       tape += key;
216       headPosition = tape.length() - 1;
217       displayTape();
218     }
219   }
220 }
```

Listing 1 – Código da maquina de Turing para Arduino

0.3.3 Estrutura do Hardware

Abaixo temos a forma de como foi montado:

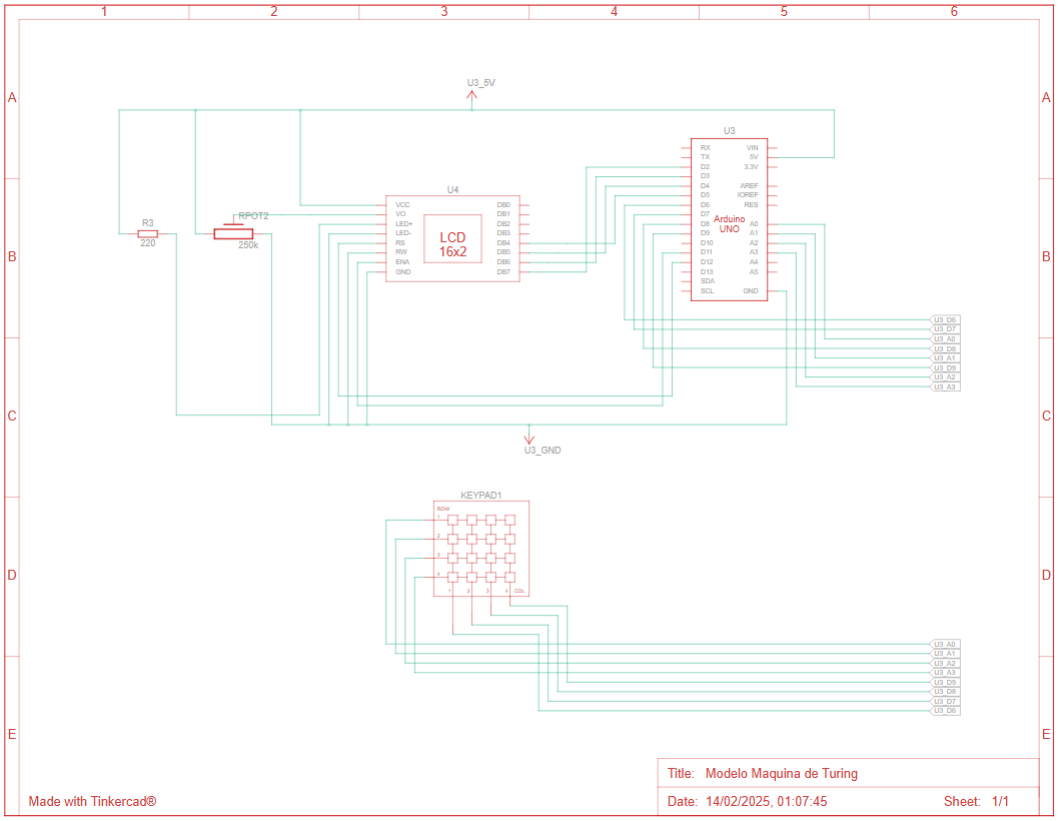

Figura 1 – Esquema Arduino

0.4 Desenvolvimento e Implementação

Nesta seção, descrevemos o fluxo do programa, os desafios encontrados e as soluções implementadas para simular o funcionamento de uma Máquina de Turing. O sistema permite que o usuário insira regras de produção, que são processadas para atualizar o estado da fita e do cabeçote. As regras são enviadas em uma linha só pelo serial monitor na seguinte sequência:**initialState:A;finalState:C;A,0,B,1,D;B,1,C,0,E;**

Onde **initialState** se refere ao estado inicial, **finalState** é o estado final, os demais símbolos são as regras de transição que são recebidos como uma quíntupla na seguinte ordem **(estado inicial,lê, para qual estado vai, escreve, para qual lado o cabeçote vai)** sendo que o programa é capaz de receber até 20 regras de transição.

Foram implementados apenas dois autômatos, que são apresentados a seguir com suas regras de formação.

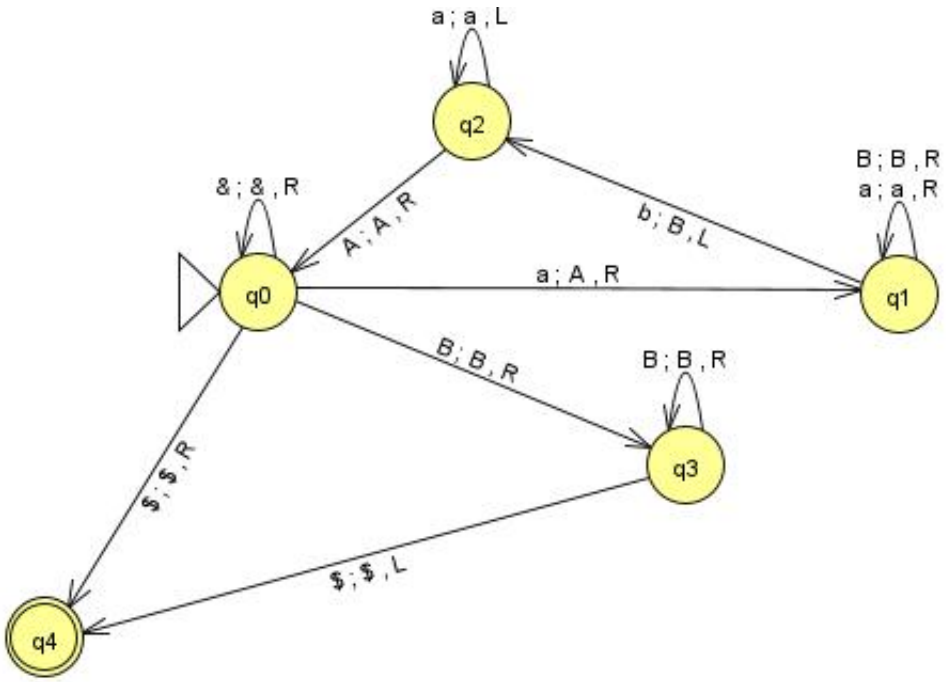

Figura 2 – Automato - 1

Regra de formação :

initialState:0;finalState:4;0,, 4,,D;0,a,1,A,D;0,,0,,D;0,B,3,B,D;1,a,1,a,D;1,b,2,B,E;
1,B,1,B,D;2,A,0,A,D;2,a,2,a,E;2,B,2,B,E;3,B,3,B,D;3,, 4,,E;4,0,4,0,E;

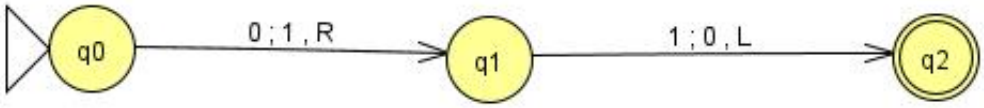

Figura 3 – Automato - 2

Regra de formação :
initialState:A;finalState:C;A,0,B,1,D;B,1,C,0,E;

0.5 Resultados e Discussão

Os testes realizados demonstraram que o sistema é capaz de interpretar e executar as regras de produção corretamente. A ideia inicial era de implementar em um esp32, criando uma interface web com html, e depois disso enviar um arquivo .json ou .txt com as regras de transição, porém não foi possível devido eu não possuir um modulo que acrescenta um slot de cartão sd. Com isso pensei de uma maneira mais fácil onde passaria as regras em uma linha só onde apenas o Arduíno iria interpretar e já processar. Para um possível upgrade, seria necessário o acréscimo de um cartão sd e substituindo por um esp32, seria possível melhorar a interação e a praticidade com um servidor web.

0.6 Conclusão

Este trabalho demonstrou a viabilidade de implementar uma Máquina de Turing utilizando Arduino, destacando a importância dos conceitos teóricos da computação aplicados em sistemas embarcados. Podendo ser melhorado em alguns pontos, viabilizando a interação com o usuário e otimizando o sistema para que não ocorra erros.

Referências

[1] SIPSER, Michael. **Introdução à Teoria da Computação**. Tradução da 2ª edição norte-americana. Porto Alegre: +A Educação - Cengage Learning Brasil, 2007. E-book. p. 144. ISBN 9788522108862. Disponível em: https://app.minhabiblioteca.com.br/reader/books/9788522108862/. Acesso em: 14 fev. 2025.

[2] YOUTUBE. Simulação de Máquina de Turing. Disponível em: https://youtu.be/--E0oCwH5r8?si=5FZkfhcl7NghuodE. Acesso em: 14 fev. 2025.

[3] TINKERCAD. Projeto de Máquina de Turing em Arduino. Disponível em: https://www.tinkercad.com/things/2EJGUHfVDp2. Acesso em: 14 fev. 2025.