MÁQUINA DE TURING: IMPLEMENTAÇÃO PRÁTICA

Felipe Francisco Bonilho Lorusso(2270226) [*], Luis Henrique Ferracciu Pagotto Mendes(2272016)[*], Maria Eduarda Soares Romana Silav(2408830) [*]

[*] *Universidade Tecnologica Federeal do Paraná*
Apucarana, Paraná, Brasil

Emails: felipelorusso@alunos.utfpr.edu.br, luismendes.2020@alunos.utfpr.edu.br, silva.2003@alunos.utfpr.edu.br

Resumo— O presente trabalho, refere-se à implementação prática de uma máquina de Turing para o reconhecimento de uma linguagem L, a qual é definida como um conjunto de palavras compostas por pares de 1's e 2's, em que a quantidade de 1's deve ser equivalente à quantidade de 2's na palavra. Para tanto, se fez necessário desenvolver os estados da máquina de Turing, as transições entre esses estados, o alfabeto de entrada e as regras de aceitação específicas para as palavras da linguagem L. Este processo foi de extrema importância, tendo em vista que estabelece uma base conceitual importantíssima para implementar a simulação da máquina, a qual foi realizada na plataforma Tinkercad, em que um modelo de hardware foi criado a fim de representar o funcionamento da máquina de Turing projetada. Durante a simulação, é possível observar a demonstração das operações realizadas pela máquina para reconhecer e processar palavras que pertencem à linguagem L.

Palavras-chave— Implementação prática, Máquina de Turing, Linguagem L, Simulação

1 INTRODUÇÃO

Em 1936, o matemático Alan Turing desenvolveu uma maquina abstrata, conhecida como Máquina de Turing, a mesma desempenha um papel crucial na teoria da computabilidade e da complexidade computacional.

A máquina é construída por uma fita infinita, decomposta em células, onde cada uma que contém caracteres de um alfabeto finito gravado, uma cabeça móvel de leitura e escrita (cabeçote), que desempenha o papel de ler e escrever símbolos na fita, podendo mover-se para a direita ou para a esquerda e um conjunto de estados que define as operações a serem realizadas de acordo com o símbolos lidos(Sipser, M.,2006).

Afim de compreender de forma prática os princípios teóricos que regem a computação por meio de máquinas abstratas, foi realizada a implementação de uma Máquina de Turing para uma linguagem específica na plataforma Tikercad, que, mesmo por uma simulação, oferece uma abordagem prática e tangível para explorar os conceitos, garantindo a compreensão profunda da teoria.

2 DEFINIÇÃO FORMAL DE UMA *MÁQUINA DE TURING*

Uma máquina de Turing é definida por uma 7-upla, $(Q, \Sigma, \Gamma, \delta, q_0, q_{aceita}, q_{rejeita})$, onde Q, Σ, Γ são todos conjuntos finitos e

1. Q é o conjunto de estados,
2. Σ é o alfabeto de entrada sem o símbolo em branco $\sqcup$,
3. Γ é o alfabeto de fita, onde $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$,
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{E, D\}$ é a função de transição,
5. $q_0 \in Q$ é o estado inicial,
6. $q_{aceita} \in Q$ é o estado de aceitação, e
7. $q_{rejeita} \in Q$ é o estado de rejeição, onde $q_{rejeita} \neq q_{aceita}$.

3 LINGUAGEM IMPLEMENTADA

Para o desenvolvimento dessa máquina de Turing a linguagem escolhida foi:

$$L = \{w \mid w \in (1 \mid 2)^* \text{ e w tem a mesma quantidade de 1's e 2's na palavra}\}$$

A linguagem L é definida como w, em que a mesma deve ser formada por pares de 1's e 2's. O * qualifica que a palavra pode ser vazia ou ter mais repetições dos caracteres. Sendo assim $(1 \mid 2)^*$, indica que a palavra deve ter uma combinação de 1's e 2's em qualquer ordem e quantidade.

Entretanto, existe uma condição para que uma palavra w pertença à linguagem L, sendo esta a necessidade de haver número de ocorrências de 1's exatamente igual ao número de ocorrências de 2's. Ou seja, para uma palavra ser reconhecida pela linguagem, é necessário que a mesma tenha uma quantidade semelhante de 1's e 2's, sem que um caractere seja mais recorrente que outro.

Essa definição formal é fundamental para compreender o tipo de palavra que é aceita pela linguagem L, sendo, portanto, a base para o desenvolvimento prático de uma Máquina de Turing capaz de reconhecer essa linguagem.

3.1 Definição Formal da Linguagem Implementada

Para a linguagem L, utilizamos uma máquina de Turing, definida pela 7-upla $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{aceita}}, q_{\text{rejeita}})$, onde:

1. Q é o conjunto de estados: $\{q_0, q_1, q_2, q_3, q_4, q_{\text{aceita}}, q_{\text{rejeita}}\}$
2. Σ é o alfabeto de entrada: $\{1, 2\}$
3. Γ é o alfabeto da fita: $\{1, 2, \sqcup, X, Y\}$
   - $\sqcup$: símbolo em branco.
   - X: marcador para '1' processado.
   - Y: marcador para '2' processado.
4. δ é a função de transição:

$$\begin{aligned}
\delta(q_0, \sqcup) &= (q_{\text{aceita}}, \sqcup, E) \\
\delta(q_0, 1) &= (q_1, X, D) \\
\delta(q_0, 2) &= (q_4, Y, D) \\
\delta(q_0, X) &= (q_0, X, D) \\
\delta(q_0, Y) &= (q_0, Y, D) \\
\delta(q_1, 1) &= (q_1, 1, D) \\
\delta(q_1, 2) &= (q_2, Y, E) \\
\delta(q_1, Y) &= (q_1, Y, D) \\
\delta(q_2, 1) &= (q_2, 1, E) \\
\delta(q_2, X) &= (q_0, X, D) \\
\delta(q_2, Y) &= (q_2, Y, E) \\
\delta(q_4, 1) &= (q_3, X, E) \\
\delta(q_4, 2) &= (q_4, 2, D) \\
\delta(q_4, X) &= (q_4, X, D) \\
\delta(q_3, 2) &= (q_3, 2, E) \\
\delta(q_3, X) &= (q_3, X, E) \\
\delta(q_3, Y) &= (q_0, Y, D)
\end{aligned}$$

5. q_0 é o estado inicial.
6. q_{aceita} é o estado de aceitação.
7. q_{rejeita} é o conjunto: $\{q_0, q_1, q_2, q_3, q_4\}$.

3.2 Diagrama de estados

Um diagrama de estados consiste em uma representação visual de uma máquina de estados finita. A figura 1, representa este diagrama, também conhecido como diagrama de transição de estados, para a linguagem L.

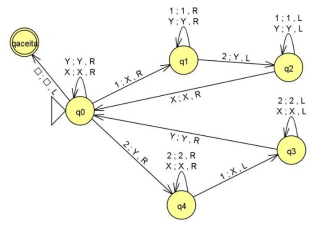

Figura 1: Diagrama de estados da linguagem L

Este diagrama (Figura 1) apresenta, de forma detalhada, as transições entre os estados e as ações que a máquina de Turing realiza, levando em consideração os símbolos que etão gravados na fita, sendo, portanto, uma forma visual e eficiente de entender o comportamento da máquina e os passos que ela toma para processar a fita de entrada.

3.3 Estados

- q_0, q_1, q_2, q_3, q_4 são os estados da máquina de Turing.
- q_{aceita} é o estado de aceitação.
- Os estados q_0, q_1, q_2, q_3, q_4 também são considerados estados de rejeição. Se a máquina não alcançar o estado de aceitação q_{aceita} e permanecer em qualquer um desses estados, a palavra é rejeitada.

3.4 Transições

Cada transição é rotulada da seguinte forma: *leitura ; escrita , movimento*, onde:

- *leitura* é o símbolo lido na fita.
- *escrita* é o símbolo escrito na fita.
- *movimento* indica a direção em que a fita se move: R (direita) ou L (esquerda).

3.5 Descrição das Transições

3.5.1 Do estado q_0:

- Lê Y, escreve Y move o cabeçote para a direita, ficando em q_0.
- Lê X, escreve X move o cabeçote para a direita, ficando em q_0.
- Lê 1, escreve X move o cabeçote para a direita, indo para q_1.
- Lê 2, escreve Y move o cabeçote para a direita, indo para q_4.
- Lê um espaço em branco ($\sqcup$), escreve $\sqcup$ move o cabeçote para a, indo para q_{aceita}.

3.5.2 Do estado q_1:

- Lê 1, escreve 1 move o cabeçote para a direita, permanecendo em q_1.
- Lê Y, escreve Y move o cabeçote para a direita, permanecendo em q_1.
- Lê 2, escreve Y move o cabeçote para a esquerda, indo para q_2.

3.5.3 Do estado q_2:

- Lê 1, escreve 1 move o cabeçote para a esquerda, permanecendo em q_2.
- Lê Y, escreve Y move o cabeçote para a esquerda, permanecendo em q_2.
- Lê X, escreve X move o cabeçote para a direita, voltando para q_0.

3.5.4 Do estado q_3:

- Lê 2, escreve 2 move o cabeçote para a esquerda, permanecendo em q_3.
- Lê X, escreve X move o cabeçote para a esquerda, permanecendo em q_3.
- Lê Y, escreve Y move o cabeçote para a direita, indo para q_0.

3.5.5 Do estado q_4:

- Lê 1, escreve X move o cabeçote para a esquerda, voltando para q_0.
- Lê 2, escreve 2 move o cabeçote para a direita, permanecendo em q_4.
- Lê X, escreve X move o cabeçote para a direita, permanecendo em q_4.

4 PROCESSO DE CONSTRUÇÃO DO SIMULADOR

Para a construção do simulador, foi utilizado o software online Tinkercad, amplamente usado para a criação de simuladores de circuitos elétricos. No nosso experimento, o Tinkercad foi essencial para projetarmos um circuito composto por um teclado matricial, um painel de LED 16x2 e um Arduino UNO.

O teclado e o painel de LED foram conectados ao Arduino UNO, que gerencia ambos os componentes. O código necessário para o funcionamento do simulador foi armazenado e executado pelo Arduino, permitindo o controle e a interação com o teclado e o painel de LED de maneira eficiente.

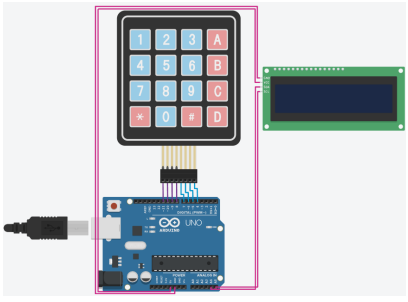

Figura 2: Circuito montado

4.1 Configuração das Conexões no Circuito

Para realizar a conexão do teclado matricial ao Arduino Uno, o seguinte esquema foi utilizado:

4.1.1 Linhas do Teclado:

- A linha 1 do teclado foi conectada ao pino digital D11 do Arduino.
- A linha 2 foi conectada ao pino digital D10.
- A linha 3 foi conectada ao pino digital D9.
- A linha 4 foi conectada ao pino digital D8.

Essas conexões estão representadas na cor roxa na Figura 2.

4.1.2 Coluna do teclado:

- A coluna 1 do teclado foi conectada ao pino digital D7 do Arduino.
- A coluna 2 foi conectada ao pino digital D6.
- A coluna 3 foi conectada ao pino digital D5.
- A coluna 4 foi conectada ao pino digital D4.

Essas conexões estão representadas na cor azul na Figura 2.

Tal configuração permite a leitura eficiente do estado de cada tecla, possibilitando a identificação precisa das entradas do usuário.

4.2 Conexão do Arduino ao Display LCD

- O pino 5V do Arduino foi ligado ao pino VCC do display LCD.
- O pino GND do Arduino foi conectado ao pino GND do display LCD.
- A porta A4 do Arduino foi conectada ao pino SDA do display LCD.
- A porta A5 do Arduino foi conectada ao pino SCL do display LCD.

Essa configuração, representada pela cor vermelha na figura 2, garante a alimentação adequada do display LCD e permite a comunicação correta entre o Arduino e a tela. Com isso, é possível enviar comandos e dados para o display, permitindo a exibição de informações conforme necessário no projeto.

5 IMPLEMENTAÇÃO NO ARDUINO

O código recebe uma entrada do usuário através do teclado matricial (configuração do hardware abordada no item 4 deste artigo), processa a sequência de acordo com os estados da máquina de Turing e exibr os resultados e as transições no display LCD.

5.1 EXPLICAÇÃO DO CÓDIGO

5.1.1 Componentes e Bibliotecas

1. **Bibliotecas incluídas:**
   - `Keypad.h`:
     Biblioteca para manipulação do teclado matricial.
   - `Wire.h`:
     Biblioteca para comunicação I2C.
   - `Adafruit_LiquidCrystal.h`:
     Biblioteca para controle do display LCD via I2C.
2. **Definição de estados:**
   - É definido um "enum `Estado`", que contém os estados da máquina de Turing, conforme a definição formal da mesma (`q0`, `q1`, `q2`, `q3`, `q4`, `qaceita`).
3. **Inicialização de variáveis e componentes:**
   - Configuração do display LCD.
   - Definição das dimensões do teclado (4x4) e mapeamento das teclas.
   - Variáveis para armazenar a palavra inserida pelo usuario (`palavra`) e o índice atual (`cabecote`).

5.1.2 Função `setup()`

- Inicializa o display LCD e exibe mensagens iniciais:
  "Máquina de Turing"e depois
  "L=(1 | 2)* nmr de 1's e2's iguais"
- Configura a comunicação serial.
- Configura as colunas do teclado com resistores pull-up internos.

5.1.3 Função `loop()`

- Exibe a mensagem "Digite a palavra"e a palavra atual no display.
- Lê os valores inseridos pelas teclas do teclado matricial, atualiza a palavra, ou processa a palavra quando a tecla '5' é pressionada.

5.1.4 Função `processarPalavra()`

1. **Inicialização:**
   - Reinicia o estado atual para `q0`.
   - Define a posição inicial do cabeçote (`cabecote = 0`).
2. **Loop Principal da Máquina de Turing:**
   (a) **Leitura do Símbolo:** Lê o símbolo atual na posição do cabeçote.
   (b) **Exibição do Estado Atual e Fita:** Atualiza o display LCD com o estado atual e a fita, mostrando a posição do cabeçote entre colchetes.
   (c) **Transições de Estado:** Dependendo do estado atual e do símbolo lido, atualiza o estado, modifica a fita e move o cabeçote.
   (d) **Finalização:** Se o estado final for `q0` e o cabeçote tiver percorrido toda a palavra, a palavra é aceita. Caso contrário, a palavra é rejeitada.
   (e) **Reinicialização ou Fim:** Pergunta ao usuário se deseja processar outra palavra ou terminar o programa.

6 PROCEDIMENTOS PARA TESTAR O SIMULADOR

O funcionamento da máquina de Turing pode ser representado a partir do passo a passo a seguir. As imagens deste passo a passo estão no Apêndice 2 deste artigo:

1. **Máquina de Turing:** Reapresentação da máquina de Turing antes da simulação (Figura 3)
2. **Iniciar a simulação:** Na plataforma Tinkercad há um botão que inicia a simulação do projeto (Figura 4).
3. **Mensagem de boas vinda:** Ao iniciar a simulação o display LCD irá mostrar duas mensagens de boas vindas: "Maquina de Turing"(Figura 5) e L = (1 | 2)* nmr de 1's e 2's iguais (Figura 6)
4. **Inserir a palavra:** Nesta etapa o usuário insere a palavra que quer testar na máquina de Turing (Figura 7)

6.1 Exemplo Prático palavra aceita:

1. **Inserir palavra:** Para realizar o exemplo prático de uma palavra aceita, foi inserida, no simulador, a palavra "12", que contém a mesma quantidade de 1's e 2's, sendo, portanto, aceita pela máquina de Turing(Figura 8).
2. **Palavra na fita:** Nesta etapa, a palavra inserida pelo usuário é gravada na fita ao clicar no botão 5 do teclado matricial, para iniciar as verificações e transições (Figura 9).
3. **Transições:** A palavra começa a ser testada de acordo com os estados de transições da Máquina de Turing (Figuras 10, 11 e 12).
4. **Palavra Aceita:** Após passar por todos os estados e ser verificada, a palavra "12"é aceita pela Máquina de Turing (Figura 13).

5. **Outra palavra ?:** Finalizando a verificação, o simulador pergunta ao usuário se quer testar outra palavra (Figura 14).

6. **Monitor Serial:** A cada etapa de verificação o monitor serial exibe uma quintupla da verificação da palavra ”12”(e, i, i’, e’, s), onde:

   - e - Estado atual
   - i - Símbolo lido
   - i’ - Símbolo a imprimir
   - e’ - Próximo estado
   - s - Sentido do movimento

   (Figura 15)

6.2 Exemplo Prático palavra rejeitada: Para realizar a verificação de uma palavra rejeitada, no passo anterior, o qual o simulador questiona se o usuário deseja inserir outra palavra, a resposta foi ”sim”. Logo, foi inserida a palavra ”122”para verificação.

1. **Inserir palavra:** Para realizar a verificação de uma palavra rejeitada, foi digitada, no teclado matricial, a palavra ”122”, que contém quantidades diferentes de 1’s e 2’s. Sendo, portando, rejeitada pela máquina de Turing (Figura 16).

2. **Palavra na fita:** Nesta etapa, a palavra inserida pelo usuário é gravada na fita ao selecionar o botão 5 do teclado matricial, para iniciar as verificações e transições (Figura 17).

3. **Transições:** A palavra começa a ser testada de acordo com os estados de transições da máquina de Turing (Figuras 18, 19,20, 21).

4. **Palavra Rejeitada:** Após passar por todos os estados e ser verificada, a palavra ”122”é rejeitada pela máquina de Turing (Figura 22).

5. **Outra palavra ?:** Mais uma vez, ao finalizar a verificação, o simulador pergunta ao usuário se quer testar outra palavra (Figura 14).

6. **Monitor Serial:** A cada etapa de verificação o monitor serial exibe uma quintupla da verificação da palavra ”122”(e, i, i’, e’, s), onde:

   - e - Estado atual
   - i - Símbolo lido
   - i’ - Símbolo a imprimir
   - e’ - Próximo estado
   - s - Sentido do movimento

   (Figura 23)

7. **Fim do programa:** Quando é perguntado ao usuário se ele deseja inserir uma nova palavra e o mesmo responder que não, o programa finaliza (Figura 24)

7 CONCLUSÃO

O presente artigo, refere-se ao desenvolvimento de uma Máquina de Turing específica para a linguagem

$$L = \{w \mid w \in (1 \mid 2)^* \text{ e w tem a mesma quantidade de 1's e 2's na palavra}\}$$

, foi demonstrada sua implementação prática através de uma simulação na plataforma online Tinkercad, específica para simulações de circuitos elétricos. O processo de desenvolvimento incluiu a definição formal da Máquina de Turing para a linguem implementada, a fim de processar e reconhecer a linguagem L de maneira eficaz.

A simulação no Tinkercad garantiu a verificação e validação da funcionalidade e eficiência da Máquina de Turing projetada, de forma visual, tendo em vista que é possível verificar o comportamento da máquina em tempo real. Os resultados obtidos na simulação confirmaram que a máquina é capaz de processar corretamente a linguagem

$$L = \{w \mid w \in (1 \mid 2)^* \text{ e w tem a mesma quantidade de 1's e 2's na palavra}\}$$

, seguindo as regras definidas e produzindo as saídas esperadas.

Além do mais, a utilização do Tinkercad como ferramenta de simulação se fez importantíssima no processo de testes e ajustes da máquina, pois fornece uma interface intuitiva. Através desta abordagem, foi possível identificar e corrigir eventuais incoerências no modelo teórico, garantindo o vigor da implementação.

Em suma, este estudo não apenas contribui para a compreensão e aplicação prática de uma Máquina de Turing para uma linguagem especifica, mas também mostra a eficácia do Tinkercad como plataforma de simulação, não apenas de circuitos elétricos, que é a forma mais usada, mas também para projetos de computação teórica. Futuras pesquisas podem expandir este trabalho para outras linguagens e aprimorando a complexidade da máquina desenvolvida, além de investigar a possibilidade da realização da simulação em outras plataformas.

Referências

Sipser, M. (2006). Introduction to the Theory of Computation. Cengage Learning.

Apêndice 1

```
#include <Keypad.h>
#include <Wire.h>  // Biblioteca para
   ↪ Comunica  o I2C
#include <Adafruit_LiquidCrystal.h>
```

```
using namespace std;
// Construtor para comunica  o I2C (
    ↪ endere o 0x20 no caso)
Adafruit_LiquidCrystal lcd(0);

enum Estado {
  q0,
  q1,
  q2,
  q3,
  q4,
  qaceita
};

Estado estadoAtual = q0;
Estado estadoAtualPrint;

const byte ROWS = 4; // four rows
const byte COLS = 4; // four columns
char keys[ROWS][COLS] = {
  {'1','2', '3', 'A'},
  {'4','5', '6', 'B'},
  {'7','8', '9', 'C'},
  {'*','0', '#', 'D'}
};
byte rowPins[ROWS] = {11, 10, 9, 8};
    ↪ // connect to the row pinouts
    ↪ of the keypad
byte colPins[COLS] = {7, 6, 5, 4}; //
    ↪ connect to the column pinouts
    ↪ of the keypad

Keypad keypad = Keypad(makeKeymap(keys
    ↪ ), rowPins, colPins, ROWS, COLS
    ↪ );

char palavra[16];
int index = 0;

void setup() {
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("   Maquina de");
  lcd.setCursor(0, 1);
  lcd.print("     Turing");
  delay(2000);
  lcd.clear();
  lcd.print("L=(1|2)* nmr de");
  lcd.setCursor(0, 1);
  lcd.print("1's e 2's iguais");
  delay(3000);
  lcd.clear();
  Serial.begin(9600);

  // Ativar pull-up internos para as
      ↪ colunas
  for (byte i = 0; i < COLS; i++) {
    pinMode(colPins[i], INPUT_PULLUP);
  }
}

void loop() {
  lcd.setCursor(0, 0);
  lcd.print("Digite a palavra");
  lcd.setCursor(0, 1);
  delay(10);
  lcd.print(palavra); // Display
      ↪ current input

  char key = keypad.getKey();
  if (key && (key =='1' || key == '2'
      ↪ || key == '5')) {
    if (key == '5') {
      palavra[index] = '\0';
      processarPalavra();
      index = 0; // Reset index for
          ↪ the next word
      memset(palavra, 0, sizeof(
          ↪ palavra)); // Clear the
          ↪ palavra array
      lcd.clear();
    } else {
      if (index < 16) { // Ensure we
          ↪ don't overflow the array
        palavra[index] = key;
        index++;
        lcd.setCursor(0, 1);
        lcd.print(palavra); // Update
            ↪ display with current
            ↪ input
      }
    }
  }
}

void processarPalavra() {
  if (strlen(palavra) == 0) { //
      ↪ Verifica se a palavra est
      ↪ vazia
    lcd.clear();
    lcd.print("Fita: vazia");
    delay(2000);
  }else{
    lcd.clear();
    lcd.print("Fita: ");
    lcd.print(palavra);
    delay(2000);  // Delay to show the
        ↪  word on the LCD
  }
  estadoAtual = q0; // Reiniciar o
      ↪ estado
  int cabecote = 0;
  lcd.clear();
  char lado;
  while (estadoAtual != qaceita &&
      ↪ cabecote >= 0 && cabecote <
      ↪ strlen(palavra)) {
    char simboloLido = palavra[
        ↪ cabecote];
    delay(1000);

    lcd.setCursor(0, 0);
    lcd.print("Estado: q");
    lcd.print((int)estadoAtual); //
        ↪ Exibe o estado como n mero
        ↪  (q0, q1, etc.)
    lcd.setCursor(0, 1);
    lcd.print("Fita: ");
```

```
    for (int i = 0; i < strlen(palavra
        ↪ ); i++) {
      if (i == cabecote) {
        lcd.print("[");  // Delimita o
            ↪  cabe ote
        lcd.print(palavra[i]);
        lcd.print("]");
      } else {
        lcd.print(palavra[i]);
      }
    }

    switch (estadoAtual) {
      case q0:
        estadoAtualPrint = q0;
        if (simboloLido == '1') {
          estadoAtualPrint = q0;
          estadoAtual = q1;
          palavra[cabecote] = 'X';
          cabecote++;
          lado = 'R';
        } else if (simboloLido == '2')
            ↪  {
          estadoAtualPrint = q0;
          estadoAtual = q4;
          palavra[cabecote] = 'Y';
          cabecote++;
          lado = 'R';
        } else if (simboloLido == 'X'
            ↪ || simboloLido == 'Y')
            ↪ {
          estadoAtualPrint = q0;
          cabecote++;
          lado = 'R';
        } else {
          lcd.clear();
          lcd.print("Palavra rejeitada
              ↪ :");
          lcd.setCursor(0, 1);
          lcd.print(palavra);
          delay(3000);  // Delay to
              ↪ show the rejection
              ↪ message
          return;
        }
        break;
      case q1:
        estadoAtualPrint = q1;
        if (simboloLido == '1' ||
            ↪ simboloLido == 'Y') {
          cabecote++;
          lado = 'R';
        } else if (simboloLido == '2')
            ↪  {
          estadoAtualPrint = q1;
          estadoAtual = q2;
          palavra[cabecote] = 'Y';
          cabecote--;
          lado = 'L';
        } else {
          lcd.clear();
          lcd.print("Palavra rejeitada
              ↪ :");
          lcd.setCursor(0, 1);
          lcd.print(palavra);
          delay(3000);  // Delay to
              ↪ show the rejection
              ↪ message
          return;
        }
        break;
      case q2:
        estadoAtualPrint = q2;
        if (simboloLido == '1' ||
            ↪ simboloLido == 'Y') {
          cabecote--;
          lado = 'L';
        } else if (simboloLido == 'X')
            ↪  {
          estadoAtualPrint = q2;
          estadoAtual = q0;
          cabecote++;
          lado = 'R';
        } else {
          lcd.clear();
          lcd.print("Palavra rejeitada
              ↪ :");
          lcd.setCursor(0, 1);
          lcd.print(palavra);
          delay(3000);  // Delay to
              ↪ show the rejection
              ↪ message
          return;
        }
        break;
      case q3:
        estadoAtualPrint = q3;
        if (simboloLido == '2' ||
            ↪ simboloLido == 'X') {
          cabecote--;
          lado = 'L';
        } else if (simboloLido == 'Y')
            ↪  {
          estadoAtualPrint = q3;
          estadoAtual = q0;
          cabecote++;
          lado = 'R';
        } else {
          lcd.clear();
          lcd.print("Palavra rejeitada
              ↪ :");
          lcd.setCursor(0, 1);
          lcd.print(palavra);
          delay(3000);  // Delay to
              ↪ show the rejection
              ↪ message
          return;
        }
        break;
      case q4:
        estadoAtualPrint = q4;
        if (simboloLido == '1') {
          estadoAtualPrint = q4;
          estadoAtual = q3;
          palavra[cabecote] = 'X';
          cabecote--;
          lado = 'L';
        } else if (simboloLido == '2'
            ↪ || simboloLido == 'X')
            ↪ {
```

```
          cabecote++;
          lado = 'R';
        } else {
          lcd.clear();
          lcd.print("Palavra rejeitada
              ↪ :");
          lcd.setCursor(0, 1);
          lcd.print(palavra);
          delay(3000);  // Delay to
              ↪ show the rejection
              ↪ message
          return;
        }
        break;
        default:
        estadoAtual = q0;
        break;

    }//fim switch

    Serial.print("\n\n(e,i,i',e',s)");
    Serial.print("\n\n onde:");
    Serial.print("\n\n e - Estado
        ↪ atual: ");
    Serial.print(estadoAtualPrint);
    Serial.print("\n i - Simbolo lido:
        ↪  ");
    Serial.print(simboloLido);
    Serial.print("\n i' - Simbolo a
        ↪ imprimir: ");
    Serial.print(palavra[cabecote]);
    Serial.print("\n e' - Proximo
        ↪ estado: ");
    Serial.print(estadoAtual);
    Serial.print("\n s - Sentido
        ↪ Movimento: ");
    Serial.print(lado);

}//fim while

if (estadoAtual == q0 && cabecote ==
    ↪  strlen(palavra)) {
  lcd.clear();
  if(strlen(palavra) == 0){
    lcd.print("Palavra aceita:");
    lcd.setCursor(0, 1);
    lcd.print("Vazia");
    delay(3000);
  }else{
    lcd.print("Palavra aceita:");
    lcd.setCursor(0, 1);
    lcd.print(palavra);
    delay(3000);  // Delay to show
        ↪ the acceptance message
  }
}else{
  lcd.clear();
  lcd.print("Palavra rejeitada:");
  lcd.setCursor(0, 1);
  lcd.print(palavra);
  delay(3000);  // Delay to show the
      ↪  rejection message
}
lcd.clear();
lcd.print("Outra palavra?");
  lcd.setCursor(0, 1);
  lcd.print("1-Sim  2-Nao");

  char opcao;
  do {
    opcao = keypad.getKey();
  } while (opcao != '1' && opcao != '2
      ↪ '); // Aguarda at  que 1 ou
      ↪ 2 seja pressionado

  if (opcao == '1') {
    // Reinicia para uma nova palavra
    index = 0;
    memset(palavra, 0, sizeof(palavra)
        ↪ );
    lcd.clear();
  } else {
    lcd.clear();
    lcd.print("Fim do programa!");
    while (true) {
    }
  }
}
```

Apêndice 2

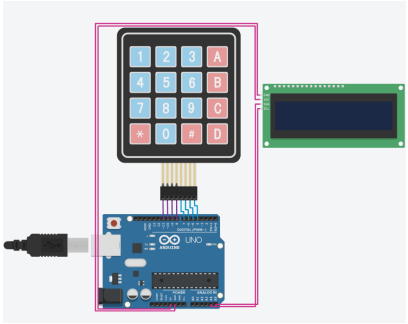

Figura 3: Maquina de Turing

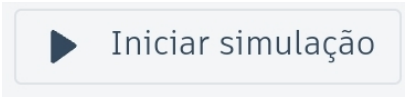

Figura 4: Botão Inicialização

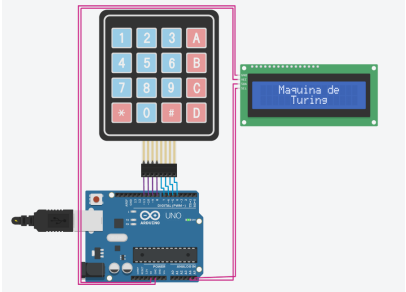

Figura 5: Inicialização da Máquina

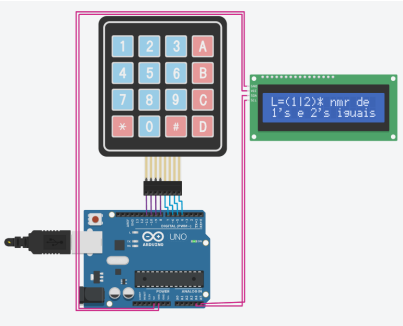

Figura 6: Print da Linguagem

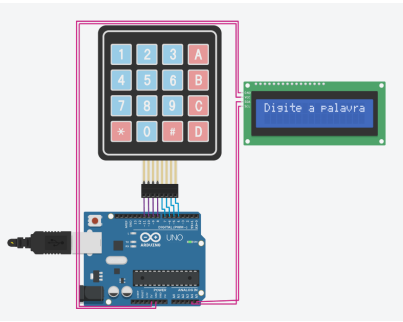

Figura 7: Digite a palavra

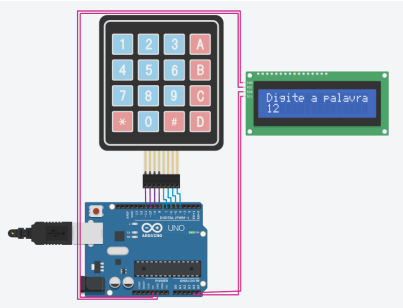

Figura 8: Palavra digitada

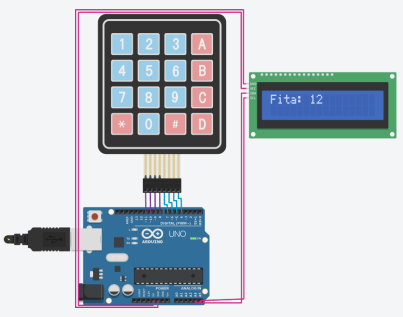

Figura 9: Fita criada

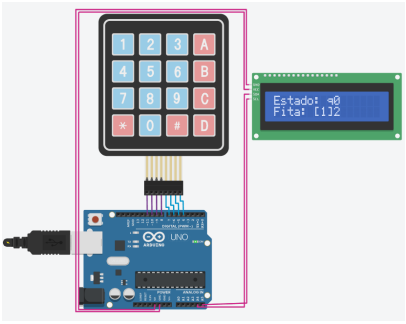

Figura 10: Algoritmo iniciado

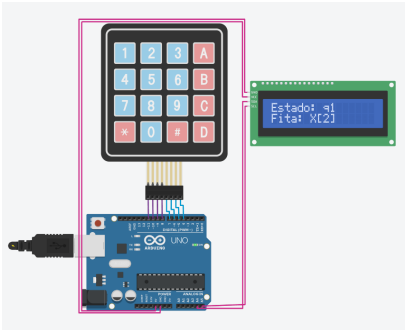

Figura 11: Algoritmo rodando

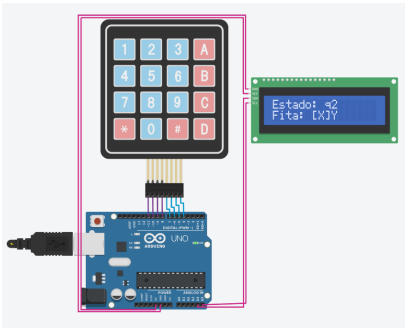

Figura 12: Algoritmo rodando

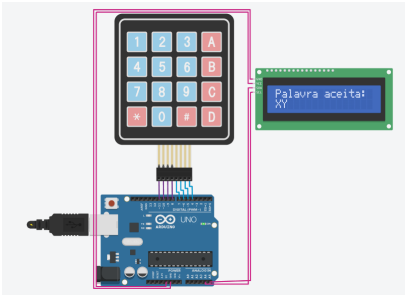

Figura 13: Palavra aceita

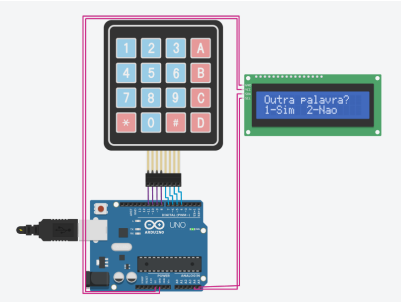

Figura 14: Menu fim do programa

```
(e,i,i',e',s)

 onde:

 e - Estado atual: 0
 i - Simbolo lido: 1
 i' - Simbolo a imprimir: 2
 e' - Proximo estado: 1
 s - Sentido Movimento: R

(e,i,i',e',s)

 onde:

 e - Estado atual: 1
 i - Simbolo lido: 2
 i' - Simbolo a imprimir: X
 e' - Proximo estado: 2
 s - Sentido Movimento: L

(e,i,i',e',s)

 onde:

 e - Estado atual: 2
 i - Simbolo lido: X
 i' - Simbolo a imprimir: Y
 e' - Proximo estado: 0
 s - Sentido Movimento: R

(e,i,i',e',s)

 onde:

 e - Estado atual: 0
 i - Simbolo lido: Y
 i' - Simbolo a imprimir:
 e' - Proximo estado: 0
 s - Sentido Movimento: R
```

Figura 15: Monitor Serial

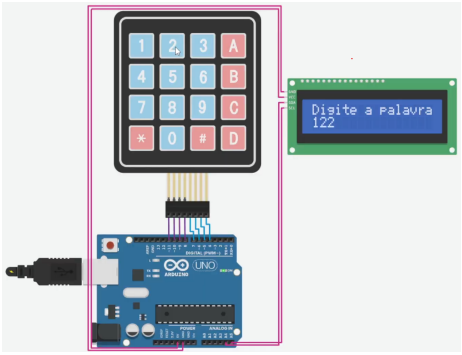

Figura 16: Palavra ”122”escrita

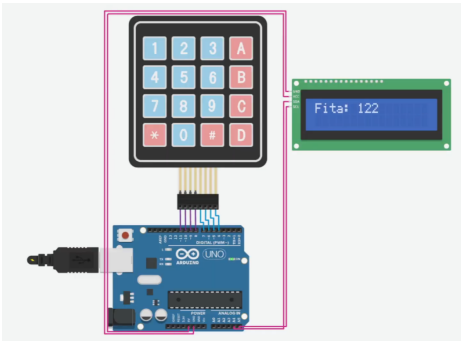

Figura 17: Fita criada

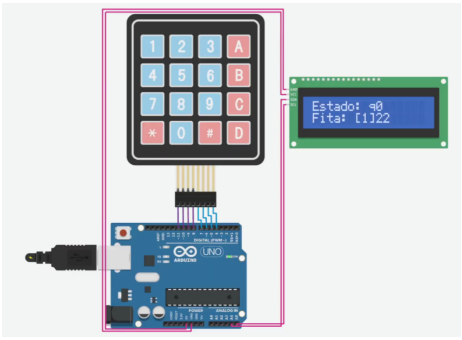

Figura 18: Inicio algoritmo para palavra ”122”

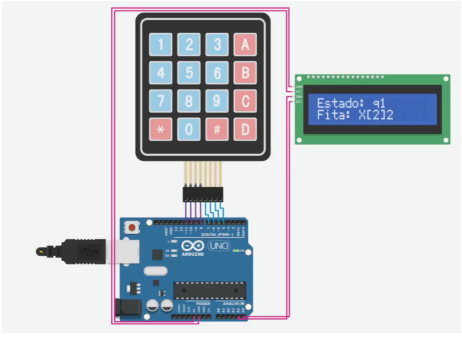

Figura 19: Algoritmo rodando

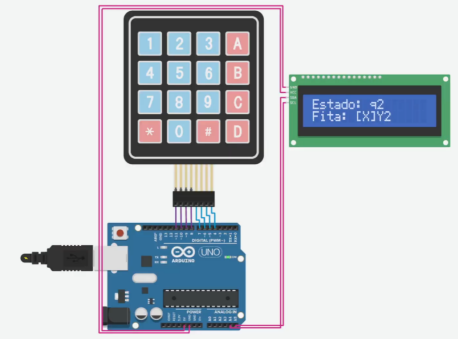

Figura 20: Algoritmo rodando

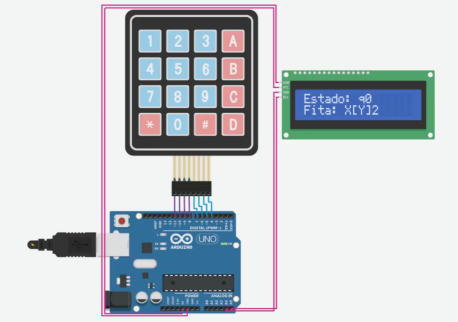

Figura 21: Algoritmo Rodando

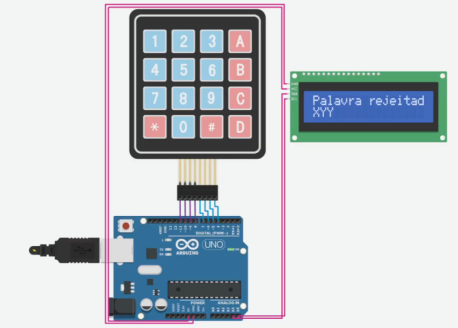

Figura 22: Print da palavra rejeitada

```
(e,i,i',e',s)

 onde:

 e - Estado atual: 0
 i - Simbolo lido: 1
 i' - Simbolo a imprimir: 2
 e' - Proximo estado: 1
 s - Sentido Movimento: R

(e,i,i',e',s)

 onde:

 e - Estado atual: 1
 i - Simbolo lido: 2
 i' - Simbolo a imprimir: X
 e' - Proximo estado: 2
 s - Sentido Movimento: L

(e,i,i',e',s)

 onde:

 e - Estado atual: 2
 i - Simbolo lido: X
 i' - Simbolo a imprimir: Y
 e' - Proximo estado: 0
 s - Sentido Movimento: R

(e,i,i',e',s)

 onde:

 e - Estado atual: 0
 i - Simbolo lido: Y
 i' - Simbolo a imprimir: 2
 e' - Proximo estado: 0
 s - Sentido Movimento: R

(e,i,i',e',s)

 onde:

 e - Estado atual: 0
 i - Simbolo lido: 2
 i' - Simbolo a imprimir:
 e' - Proximo estado: 4
 s - Sentido Movimento: R
```

Figura 23: Monitor serial palavra ”122”

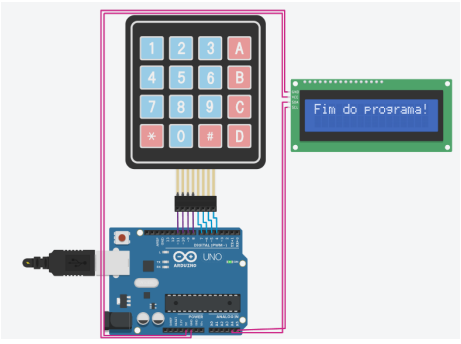

Figura 24: Fim do Programa