Simulação Tinkercad

* Sistemas Embarcados: Prof. Marco Reis - marco.reis@ba.docente.senai.brr

João Vitor Silva Mendes
Estudante de Engenharia Elétrica
SENAI - Cimatec
Camaçari, Brazil
joao.mendes@aln.senaicimatec.edu.br

***Resumo*—Neste documento será retratado o desenvolvimento de 3 protótipos, utilizando o Tinkercad para desenvolver e testar o projeto e como microcontrolador o Arduino Uno R3. O primeiro protótipo a ser desenvolvido foi a calculadora, utilizando um teclado matricial 4x4, em que principal função é resolver operações matemáticas simples, porém, também foi adicionada a função de recursividade onde o usuário pode reutilizar o resultado da operação anterior. O segundo protótipo a ser desenvolvido foi o Dispersor de Álcool automatizado. O dispersor possui um servo motor que funciona para abrir as portas de uma caixa, onde estava armazenado todo o sistema. Dentro da caixa, há um sensor de proximidade que irá detectar se a mão do usuário está dentro da caixa e então dispensar o álcool, caso não, a porta se fecharia. O último protótipo é um cofre, que possui um teclado matricial 4x4 para a entrada da senha, assim o arduino pode detectar se a sequência está correta e então se verdadeiro abrirá a porta, caso seja falso manterá trancada.**

***Abstract*—This document aims to describe the development of three prototypes in the class of Embebed Systems, it was utilized the Tinkercad for design and test the project and the microcontroller used was Arduino Uno R3. The first one is the Calculator, using a matrix keypad 4x4, the principal function is do simple math operations, and was created a recursion function to continue the operations. The second prototype is a Alcohol Dispenser, and the idea was a simples system that will detect the presence of a hand and dispense alcohol. The Dispenser have a servo motor too, to open a door for the hand, and all of the system will be inside a box. The last one it's a Safe Box, with a keyboard 4x4 for enter the password, the arduino can detect, if is correct the system can open the lock, but if the password was not correct the system will keep locked.**

***Index Terms*—Sistemas Embarcados, Arduino, Microcontroladores, Calculadora, Dispenser de Álcool, Cofre.**

I. Introdução

Durante o curso de Sistemas Embarcados, o Prof. Marco Reis propôs o desenvolvimento de três protótipos ao longo da primeira unidade. Desenvolvendo primeiro uma calculadora, em seguida um dispersor de Álcool e um cofre, O sistema utilizado para elaboração de projetos e simulações foi o Tinkercad, ferramente gratuita pertencente a AutoDesk. Além da utilização da plataforma de desenvolvimento Arduino Uno R3, que foi criado originalmente para ser uma plataforma de prototipagem open source, porém ganhou grande popularidade e atualmente é utilizada tanto por profissionais como amadores. [**?**]

II. Objetivos

Esta atividade tem como principal objetivo aplicar as teorias computacionais desenvolvidas em sala de aula, introduzir os alunos ao desenvolvimento de IOT's utilizando o arduino como microcontrolador, além de aprimorar habilidades de resolução de problemas envolvendo sistemas embarcados. [**?**]

III. Componentes Utilizados

A. Calculadora

1) Arduino Uno R3;
2) Protoboard 400 pontos;
3) Teclado Matricial 4x4;
4) Resistor 200 ;
5) Resistor 1 K.;
6) Fio Jumper(15u).

B. Dispensor de Álcool

1) Arduino Uno R3;
2) Protoboard 800 pontos;
3) Servo Motor 9g;
4) Sensor Ultrassônico;
5) Motor CC;
6) Piezo;
7) Push Boton;
8) Potenciômetro;
9) Resistor 200 (3u);
10) Capacitor 10nf;
11) Fio jumper(12u)

C. Cofre

1) Arduino Uno R3;
2) Protoboard 400 pontos;
3) Teclado Matricial 4x4;
4) Resistor 200 ;
5) Resistor 1 K.;
6) Fio Jumper(15u).

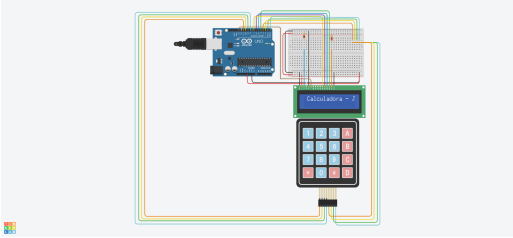

Fig. 1. Protótipo Calculadora.

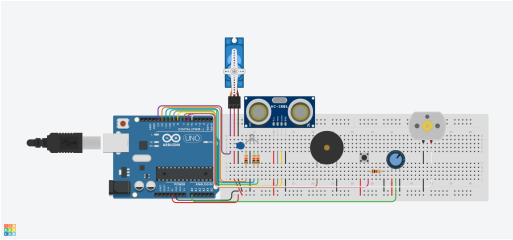

Fig. 2. Protótipo Dispensor de Álcool.

IV. Protótipos

V. Códigos

A. Calculadora

```
1  #include <Keypad.h>
2  #include <Wire.h>
3  #include <LiquidCrystal.h>
4  #include <stdio.h>
5
```

Aqui é possível analisar quais bibliotecas foram utilizadas para o desenvolvimento da calculadora. A Keypad.h é utilizada para ter acesso as funções do teclado matricial, assim como a LiquidCrystal.h para obter acesso ao pacote de funções para controlar a tela LCD. A biblioteca Wire.h tem como função garantir a interação entre o Arduino e os dispositivos conectados, já a biblioteca stdio.h foi utilizada para dar acesso as funções mais básicas da linguagem C++.

```
1   char customKey, start;
2   const byte ROWS = 4;
3   const byte COLS = 4;
4   char keys[ROWS][COLS] = {
5      {'1','2','3','+'},
6      {'4','5','6','-'},
7      {'7','8','9','*'},
8      {'C','0','=','/'}
9      };
10  byte rowPins[ROWS] = {11,10,9,8};
11  byte colPins[COLS] = {3,2,1,0};
12  Keypad customKeypad = Keypad(makeKeymap(keys), rowPins, ↩
       colPins, ROWS, COLS);
13
```

A declaração das variáveis do teclado matricial é feita da seguinte forma. Primeiro há a definição das variáveis que serão utilizadas, depois é construida uma matriz key do tipo char que será orientada pelas variáveis 'ROWS' e 'COLS', respectivamente linhas e colunas. Além disso é definido na linha 13 os pinos do teclado, baseando-se nas declarações feitas anteriormente na linha 11 e 12 que define os pinos em relação a matriz declarada na linha 4.

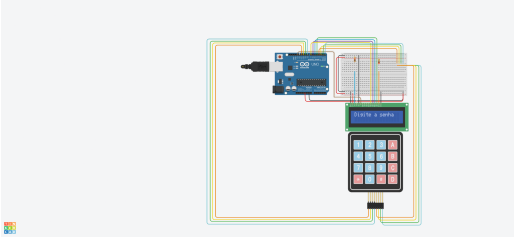

Fig. 3. Protótipo Cofre.

```
1   const int rs = 12, en = 13, d4 = 7, d5 = 6, d6 = 5, d7 = 4;
2   LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
3
```

Neste bloco de código é definido as variáveis que representaram os pinos da tela lcd 16x2. Na linha 2 essas variáveis são definidas para cada porta do LCD.

```
1   double first = 0;
2   double second = 0;
3   double third = 0;
4   double total = 0;
5
```

Neste bloco são definidas as variáveis do tipo double que representaram os números a serem cálculados durante o loop.

```
1 void setup(){
2  lcd.begin(16, 2);
3  for(int i=0;i<=20;i++){
4    lcd.setCursor(5,0);
5    lcd.print("Calculadora – Joao Vitor");
6    delay(100);
7    lcd.scrollDisplayLeft();
8    delay(100);
9    }
10 lcd.clear();
11 lcd.setCursor(0,0);
12          }
13
```

Na função setup é inicializado o LCD, após isso um loop que apresenta o título do protótipo e em seguida a função lcd.scrollDisplayLeft() para os caracteres mudarem de posição e apresentar toda a frase.

```
1 void loop(){
2   customKey = customKeypad.getKey();
3   switch(customKey){
4      case '0'...'9':
5      lcd.setCursor(0,0);
6      first = first * 10 + (customKey – '0');
7      lcd.print(first);
8      break;
9
10     case '+':
11       if(first != 0){
12          lcd.setCursor(15,1);
13          lcd.print("+");
14          second = SecondNumber();
15          total = first + second;
16          lcd.setCursor(0,3);
17          lcd.print(total);
18          first = 0;
19          second = 0;
```

```
20            }
21            else if(first == 0){
22            third = ThirdNumber(total, '+');
23            total = total + third;
24            lcd.setCursor(0,3);
25            lcd.print(total);
26            third = 0;
27            }
28      break;
29
30      case 'C':
31      total = 0;
32      first = 0, second = 0;
33      lcd.clear();
34      break;
35      }
36    }
37
```

Na função loop será o local em que os cálculos serão efetuados. No primeiro momento é utilizada a função getkey() associada a variável definida para representar o teclado é possível obter o que o usuário está digitando, e o retorno dessa função que será um char é empregado a variável 'customKey'. Logo após isso, o código segue para um switch onde é possível entrar em 6 alternativas, a primeira que representa os números de 0 a 9, a segunda até a quinta que representarão as operações caso o usuário dê a entrada no teclado(no código acima é representado apenas o caso da operação soma, com intuito de simplificar o processo já que os casos das demais operações funcionam da mesma forma), e por último o caso 'C'. No caso '0...9' representado na linha 3, tem como função formar o primeiro número que será alocado na variável 'first'. Porém, na primeira tentativa de execução o código na linha 6, possuía o seguinte formato: (first = first + customKey), e com isso apresentava na tela o código ASCII, designado para cada carácter que representava um número. Para corrigir este problema o customKey passou a subtrair o código ASCII que representava o carácter '0', para possuir o seu real valor. Para a variável first poder receber mais de um algorismo foi necessário incrementá-la de forma que o seu próprio valor fosse multiplicado por uma dezena e assim somado ao novo valor recebido pela variável 'customKey' subtraindo o valor em ASCII do carácter '0'. Após cada digito o código executa a função lcd.print() que imprimirá na tela do usuário cada algorismo digitado. Quando o usuário termina de digitar o primeiro número é esperado a entrada de uma das 4 operações, nesse exemplo será utilizada a operação da soma. Quando houver a entrada da operação soma, o código entrará no case '+' iniciado na linha 10, o programa primeiro verifica através de uma estrutura de decisão se o usuário já deu a entrada do primeiro número, se sim o usuário prosseguirá com o laço, caso não o usuário voltará ao inicio do loop, obrigando o usuário a dar a entrada de um valor. Porém, se já houver esse primeiro número. o código entrará na estrutura if e seguirá o seguinte processo: é impresso na tela nas coordenadas (15,1) como mostra a função lcd.Setcursor() a operação a ser executada. Após isso é utilizada a função SecondNumber(), o código da função segue abaixo. [**?**]

```
1         long SecondNumber(){
2   while( 1 ){
3     customKey = customKeypad.getKey();
4     if(customKey >= '0' && customKey <= '9'){
5       second = second * 10 + (customKey - '0');
6       lcd.setCursor(0,1);
7       lcd.print(second);
8       }
9
10    if(customKey == '=') break;
11          }
12  return second;
13                }
14
```

A função SecondNumber segue uma lógica parecida com a utilizada no case '0...9', mas no lugar de uma função loop é utilizada um laço de repetição while. Além disso, é utilizado outra estrutura if com intuito de verificar se algum momento será pressionada a tecla que representa o sinal '=' que iria interromper o loop. Com o fim da função ela retornará uma variável do tipo long. De volta a função loop, na linha 14 a variável 'second' recebe o valor retornado para a função, que em segui é somado com o valor de 'first' e atribuído a variável 'total', assim o valor é imprimido na segunda linha da tela com o sinal de '='. Ao final, os valores de 'first' e 'second' são zerados O programa repetirá esse mesmo processo até ser encerrado se seguir essa mesma lógica de entrada. Porém, a função ThirdNumber() foi adicionada com intuito de poder usar o valor totalizado pela operação anterior em uma nova operação, tornando a calculadora recursiva. Desse modo, se analisarmos o caso em que o usuário entra com o valor de uma operação mesmo com as variáveis 'first' e 'second' zeradas, é possível fazer a operação entre o terceiro número digitado pelo usuário com o valor totalizado anteriormente. A função ThirdNumber() é localizada na estrutura Else If, nos casos de operação dentro do switch, estes que verificam se é possível fazer a recursão explicada anteriormente.

```
1 long ThirdNumber(double r, char op){
2   lcd. clear();
3   lcd.setCursor(0,0);
4   lcd.print(r);
5   lcd.setCursor(15,1);
6   lcd.print(op);
7   while( 1 ){
8     customKey = customKeypad.getKey();
9     if(customKey >= '0' && customKey <= '9'){
10      third = third * 10 + (customKey - '0');
11      lcd.setCursor(0,1);
12      lcd.print(third);
13      }
14
15    if(customKey == '=') break;
16      }
17  return third;
18                }
19
```

Esta função segue a mesma lógica que a função SecondNumber(), porém antes de entrar no laço de repetição, o programa faz algumas alterações visuais na tela para imprimir o valor da variável 'total' representado agora pela variável 'r' e é impresso também a operação atribuída a variável 'op', estas variáveis são recebidas pela função ainda na linha 21 da função loop(). Por fim, é retornado um valor long que será recebido pela variável 'third'. O processo segue o mesmo padrão na

função loop() das demais operações feitas anteriormente.

B. Dispensor de Álcool

No código para o dispersor de álcool a única biblioteca utilizada é a Servo.h, que permite o acesso a funções para o funcionamento do servo motor.

```
1 //HEADERS
2 #include <Servo.h>
3 // SETSENSOR
4    byte pinoTransmissor = 9; // trig
5    byte pinoReceptor = 8; // echo
6    float cm ,duracao; // comprimento de onda/ tempo de resposta
7 //SET LED RGB
8    const byte R = 12;
9    const byte B = 11;
10   const byte G = 10;
11 //SET PUSH BOTON
12  const int botao = 6;
13  int estadoBotao = 0;
14 //SET SERVO MOTOR
15   int pinServo = 13;
16   Servo s;
17   int angulo = 0;
18 //MOTOR
19  int pin_motor = 5;
20  float motor;
21
```

Neste trecho de código é possível analisar as declarações de variáveis utilizadas e também dos dispositivos utilizados no sistema. A partir do comentário ”SETSENSOR”, é declarado para o sistema os pinos que são ligados ao sensor, no caso das portas trig e echo, além das variáveis ’cm’ e ’duracao’, que irão obter os valores de distância e tempo em que a sonda irá até o objeto e voltará. Em seguida é declarada as variáveis referentes ao LED RGB que está presente no sistema para representar a distância da mão do usuário até o sensor. A partir do cometário ”SETPUSHBOTON” é declarada as variáveis que representam o pino do botão e o estado do botão, respectivamente as variáveis ’botão’ e ’estadoBotao’. A partir de ”SETSERVO” é definida as variáveis do servo motor, primeiro é definido o pino do servo motor, em seguida uma variável ’s’ que representará o motor durante todo o código e um inteiro que armazenará o angulo de rotação. Por último, é definido o pino do motor é definido e uma variável ’motor’ para receber do potenciômetro a velocidade em que o motor irá descarregar o álcool.

```
1 void setup(){
2    // LED and PUSH BOTON
3    pinMode(R, OUTPUT);
4    pinMode(G, OUTPUT);
5    pinMode(B, OUTPUT);
6    // sensor
7    pinMode(pinoTransmissor, OUTPUT); // transmissor
8    pinMode(pinoReceptor, INPUT);     // receptor
9    //porta serial
10   Serial.begin(9600);
11   //servo
12   s.attach(pinServo);
13   s.write(0);
14   //boton
15 }
16
```

Na função setup em primeiro momento associa os pinos do LED RGB a saıda de dados, logo após é definido o pino transmissor como saıda de dados e o pino receptor como entrada de dados. Com isso, é iniciado o serial e definida a variável associada ao pino do servo motor para o sistema e executando a função servo.write() associada a variável ’s’ que representa o servo motor na posição ’0’, para indicar a posição em que o servo deve iniciar. Para entender a função loop é necessário antes entender algumas funções que serão utilizadas. As primeiras a serem retradas serão as funções deseginadas a orientar o funcionamento do LED RGB.

```
1  void red(){
2    analogWrite(R, 255);
3    analogWrite(G, 0);
4    analogWrite(B, 0);
5  }
6 void green(){
7    analogWrite(R, 0);
8    analogWrite(G, 255);
9    analogWrite(B, 0);
10 }
11 void blue(){
12   analogWrite(R, 0);
13   analogWrite(G, 0);
14   analogWrite(B, 255);
15 }
16 void apaga(){
17   analogWrite(R, 0);
18   analogWrite(G, 0);
19   analogWrite(B, 0);
20 }
21
```

As funções red(), green() e blue() são chamadas com intuito de acionar o LED com 3 cores diferentes que representarão a distância entre a mão do usuário e o sensor. E no caso da função apaga(), é utilizada para apagar o led quando o funcionamento do sensor for interrompido.

```
1 void gira(){
2    angulo = 250;
3    s.write(angulo);
4    delay(1000);
5
6    }
7
8 void volta(){
9    angulo = 0;
10   s.write(angulo);
11   delay(1000);
12   }
13
```

As funções gira() e volta() são utilizadas para orientar o funcionamento do servo motor, que será responsável por abrir e fechar a porta do sistema de higienização, após o giro do servo motor, há também no código uma função delay() para o sistema de higienização funcionar apenas quando o dispositivo abrir ou fechar a porta por completo.

```
1 float distancia(){
2
3  digitalWrite(pinoTransmissor, LOW);
4  delayMicroseconds(5);
5  digitalWrite(pinoTransmissor, HIGH);
6  delayMicroseconds(10);
7  digitalWrite(pinoTransmissor, LOW);
8  duracao = pulseIn(pinoReceptor, HIGH);
9  float calcDistancia= (duracao/2) * 0.0343;
10
11  return calcDistancia;
```

```
12    }
13
```

A última função a ser criada é a distancia(), que serve para fazer o cálculo da distância da mão do usuário até o sensor ultrassônico. Em primeiro momento são enviados pulsos através da função digitalWrite(), dois baixos e um alto, sendo eles intercalados, com intervalo de 5 microssegundos do primeiro para o segundo, e 10 microssegundos do segundo para o terceiro. Após isso, é chamada a função pulseIn() que recebe o tempo em que o pulso levou para ir até o objeto e voltar, este dado é armazenado na variável 'duracao'. O próximo passo é definida uma variável float, chamada 'calcDistancia' que receberá o resultado de um cálculo envolvendo o produto entre a velocidade do som(em centímetro por microssegundos) e a metade do tempo em que o pulso levou para ir ao objeto e voltar. Por fim, é retornado o valor da distância em centímetros.

```
1 void dispense(){
2    motor = analogRead(A0)/4;
3    analogWrite (pin_motor, motor);
4    delay(2000);
5    analogWrite(pin_motor, 0);
6    tone(7, 100, 200);
7    while(cm<50){
8       cm = distancia();
9       if(cm>50){
10        green();
11        delay(1000);
12        volta();
13        }
14      }
15   }
16
```

A função dispense() orienta o sistema a dispensar o álcool e como dispensar. Em primeiro momento a variável motor irá receber informações da porta analógica 'A0', que está conectada a um potenciômetro de 10K.ohm, dessa forma regulando a velocidade do motor, consequentemente a quantidade de liquido a ser dispersado. Logo após, a função analogWrite() recebe duas informações, primeiro o pino do motor e a variável 'motor' que recebeu o valor da intensidade da corrente a ser enviada para o motor CC. Durante a dispersão o código apresenta uma função delay() que irá regular também a quantidade de álcool a ser expelido pelo tempo de dispersão(Definido por padrão pelo sistema 1000ms), após esse intervalo de tempo, uma outra função analogRead() irá receber as informações para interromper o funcionamento do motor CC. Por fim, um Piezzo localizado na porta 7, é acionado pela função analogWrite(), a fim de avisar ao usuário que a dispersão já foi realizada. Durante a primeira fase de desenvolvimento, o código da função dispense() acabava nesse momento, após sair da função, havia outra função volta() que fecharia a porta do sistema, porém, neste caso o sistema poderia fechar a porta com a mão do usuário ainda dentro do recipiente. A segunda solução foi colocar um delay, mas ainda não era algo eficiente. Contudo, como o sistema possui um sensor para verificar a distância, o código passou a conter um laço de repetição que se repetiria até o momento que o sensor não verificasse mais a mão do usuário dentro do sistema, com isso permitindo que o mecanismo fechasse a porta do sistema.

```
1 void loop(){
2  estadoBotao = digitalRead(botao);
3
4  if (estadoBotao == HIGH){
5    gira();
6    blue();
7    cm =  distancia();
8    if(cm >0 && cm<=30){
9      red();
10     dispense();
11   else if (cm >50 && cm<=100) {
12       green();
13       volta();
14
15   }
16   else if (cm>100) {
17       blue();
18       volta();
19
20   }
21   else {
22       volta();
23       apaga();
24   }
25
26 }
27
```

Enfim, na função loop(), em primeiro momento é checado o estado do botão e armazenado na variável 'estadoBotao', após a verificação se o botão estiver apertado ele entre em uma estrutura if, caso não ele segue repetindo o loop. Ao entrar na estrutura, a função gira() é acionada e a porta abre e acende o led azul para indicar que o usuário pode colocar a mão. Após isso a variável 'cm' recebe a distância da mão do usuário da função distancia(), com isso uma estrutura de seleção verifica a distância e responde através do LED RGB, azul para vazio, verde para indicar que a mão do usuário ainda está muito longe e vermelho para indicar que a distância está correta e só então é dispensado o fluido através da função dispense(). Vale lembrar que dentro da função dispense existe um mecanismo para verificar se o usuário já tirou a mão e só então o mecanismo fecha a porta do sistema.

C. Cofre

```
1 #include <Keypad.h>
2 #include <Wire.h>
3 #include <LiquidCrystal.h>
4 #include <stdio.h>
5 #include <string.h>
6
```

Até a linha 4 as bibliotecas inclusas já foram discutidas neste relatório, porém, neste projeto de cofre há a adição de uma nova biblioteca, chamada de string.h com intuito de acessar funções para manipular strings. Em suma, outras declarações também já foram expostas neste relatório e não serão abordadas novamente, como a declaração do teclado matricial e do display 16x2. Sendo assim, o próximo ítem a ser abordado será a função setup() e declaração das variáveis utilizadas durante o programa.

```
1 char pass[3];
2 int cmp, cont, tentativas, chave = 1;
3
4 Keypad customKeypad = Keypad(makeKeymap(keys), rowPins, ↩
        colPins, ROWS, COLS);
5
6 void setup(){
7
8   lcd.begin(16, 2);
9   strcpy(pass, "0000");
10  tentativas = 0;
11
12 }
13
```

Na linha 1 é declarada a variável 'pass' do tipo string que terá 4 caracteres, esta armazenará a entrada do usuário. Logo, após há a declaração dos inteiros, primeiro 'cmp' que receberá o retorno da comparação entre a senha do sistema e a entrada do usuário. A variável 'cont' funcionará como um contador dentro do loop(), tentativas irá verificar quantas vezes o usuário errou a senha e se exceder o limite destruirá os dados do cofre. Por fim, a chave que irá auxiliar a interrupção do funcionamento do cofre caso o usuário exceda o limite de erros. Na função setup(), é iniciado o LCD, após isso é atribuído a variável 'pass' a string '0000', as tentativas são zeradas.

```
1 void correct(){
2   lcd.clear();
3   strcpy(pass, "0000");
4   cont = 0;
5   lcd.setCursor(0,0);
6   lcd.print("Destrancado");
7   delay(2000);
8   tentativas = 0;
9 }
10
```

A função correct() será responsável por enviar para o usuário a mensagem que o cofre está destrancado e zerar as variáveis que serão utilizadas novamente.

```
1 void incorrect(){
2   lcd.clear();
3   strcpy(pass, "0000");
4   cont = 0;
5   lcd.setCursor(0,0);
6   lcd.print("SENHA NEGADA!");
7   delay(2000);
8   lcd.clear();
9   if(tentativas == 3) autodestruir();
10 }
11
```

A função incorrect() funciona da mesma forma que a função correct(), porém envia o sinal que a senha ta negada. Porém, se o número de tentativas for excedido, a função autodestruir() é acionada.

```
1 void autodestruir(){
2   int i;
3   for(i = 3; i >= 0; i--){
4   lcd.clear();
5   lcd.setCursor(0,0);
6   lcd.print("Destruicao");
7   lcd.setCursor(0,1);
8   lcd.print("arquivos em: ");
9     lcd.print(i);
10    delay(1000);
11  }
12  chave = 0;
13 }
14
```

A função autodestruir() em primeiro momento declara um inteiro 'i' que será o contador da estrutura de repetição for a seguir. Esta estrutura servirá para iniciar a contagem regressiva e imprimir na tela para o usuário, avisando que os dados serão destruídos, apagando e imprimindo novamente a mesma mensagem, mudando apenas o valor do inteiro 'i'. Após isso a chave é zerada e o programa irá parar de funcionar.

```
1 void verifica(){
2   cmp = strcmp("2507", pass);
3   lcd.setCursor(5,0);
4   if(cmp == 0) correct();
5   else incorrect();
6   tentativas++;
7 }
8
```

A função verifica() tem como objetivo verificar se a entrada do usuário é igual a senha determinada pelo sistema, através da função strcmp. A variável 'cmp' recebe o retorno da comparação que se for positivo é igual a '0'. Logo após, uma estrutura de if analise se 'cmp' for igual a '0', é acionada a função correct(), senão é acionada a função incorrect() e o número de tentativas é incrementado o valor de uma unidade.

```
1 void loop(){
2
3   while(chave == 1){
4     lcd.setCursor(0,0);
5     lcd.print("Digite a senha");
6     customKey = customKeypad.getKey();
7
8     switch(customKey){
9
10      case '0'...'9':
11        lcd.setCursor(0,1);
12        if(cont < 4){
13          pass[cont] = customKey;
14          lcd.print(pass);
15          cont++;       }
16          else{
17          verifica();   }
18
19          break;
20
21      case 'C':
22          lcd.clear();
23          strcpy(pass, "0000");
24          cont = 0;
25          break;
26
27      case '#':
28          verifica();
29          break;
30
31
32          }
33  } if(chave == 0){
34    lcd.clear();
35    lcd.setCursor(0,0);
36    lcd.print("ERROR");
37    }
38
39  }
40
```

Na função loop(), é iniciado um laço de repetição while, para enquanto a chave for igual a '0' o programa continuará funcionando, caso não o programa irá parar de funcionar. Então, é impresso na tela LCD para o usuário a mensagem 'Digite a senha', com isso a variável 'customKey' receberá a entrada do usuário de um carácter. Com isso entra em um switch baseado na variável 'customKey', se a entrada do usuário for um número, entrará no case '0...9', a estrutura if irá verificar se o 'cont' está abaixo de quatro, que é o número máximo de caracteres por senha, com isso a variável 'pass' irá receber esses caracteres que o usuário digitar no teclado, assim que atingir o usuário der entrada de " para verificar a senha ou exceder o número limites de dígitos o programa irá entrar na função verifica(). Como dito antes, no caso da entrada ", a função verifica() será chamada, ou no caso da entrada ser igual a 'C' será zerado tanto a variável 'pass' quanto a variável 'cont' e a tela LCD será limpada. Caso a função autodestruir() seja chamada, o laço while será interrompido, e então é impresso no LCD a mensagem de "ERROR" e o programa sera quebrado.

VI. Conclusão

Baseado-se nas simulações realizadas no TinkerCAD, os experimentos concluíram todos os objetivos propostos com sucesso. A elaboração dos protótipos permitiu a aplicação da teoria computacional desenvolvida em sala de aula, além de desenvolver outras habilidades essenciais para desenvolvimento de IOT's, como as aplicações da Lei de Ohm, elaboração de circuitos elétricos simples e programação de sistemas embarcados.