



Sistema de Segurança com Verificação Formal para Indústria 4.0

Aluno: Janei Vieira
Disciplina: ECAI – IoT
Data: 05/07/2025
Sistema Completo: <https://wokwi.com/projects/435683886760102913>
Testes Unitários: <https://wokwi.com/projects/435686922316394497>
Repositório Github: <https://github.com/oMaestro174/sistema-de-seguranca-iot>



Objetivo do Projeto

Este projeto consiste no desenvolvimento de um sistema embarcado de controle de acesso para ambientes industriais, utilizando um Arduino UNO. O sistema implementa uma **máquina de estados de Mealy** para gerenciar o fluxo de autenticação via teclado matricial e detecção de movimento com um sensor MPU6050. A segurança e a robustez do sistema são aprimoradas através de testes de unidade com ArduinoUnit e verificação formal de funções críticas utilizando o C Bounded Model Checker (CBMC).



Componentes e Ligações

Arduino UNO + Periféricos:


- **Teclado 4x4:** Pinos D9 a D2 (linhas/colunas)
- **Servo Motor:** Pino D10 (sinal), 5V, GND
- **MPU6050:** Pinos A4 (SDA), A5 (SCL), 5V, GND



1. Teclado Matricial 4x4

O teclado possui 8 pinos: 4 linhas e 4 colunas. No código usamos:

Função	Pino do Teclado	Pino do Arduino UNO
Linha 1	R1	D9
Linha 2	R2	D8
Linha 3	R3	D7
Linha 4	R4	D6
Coluna 1	C1	D5
Coluna 2	C2	D4
Coluna 3	C3	D3
Coluna 4	C4	D2

 **Dica:** Ligue em sequência da esquerda para a direita no Wokwi ou Protoboard, e siga a ordem correta no código do `Keypad` .

2. Servo Motor

Fio do Servo	Conecta em
Sinal (laranja/amarelo)	D10 (pino de controle no código)
VCC (vermelho)	5V
GND (preto/marrom)	GND

 **Atenção:** Use um capacitor ou fonte externa se o servo estiver tremendo ou fraco.

3. MPU6050 (I2C)

A MPU6050 se comunica via I2C, com 4 pinos principais:

Pino MPU6050	Pino Arduino
VCC	5V
GND	GND
SDA	A4
SCL	A5


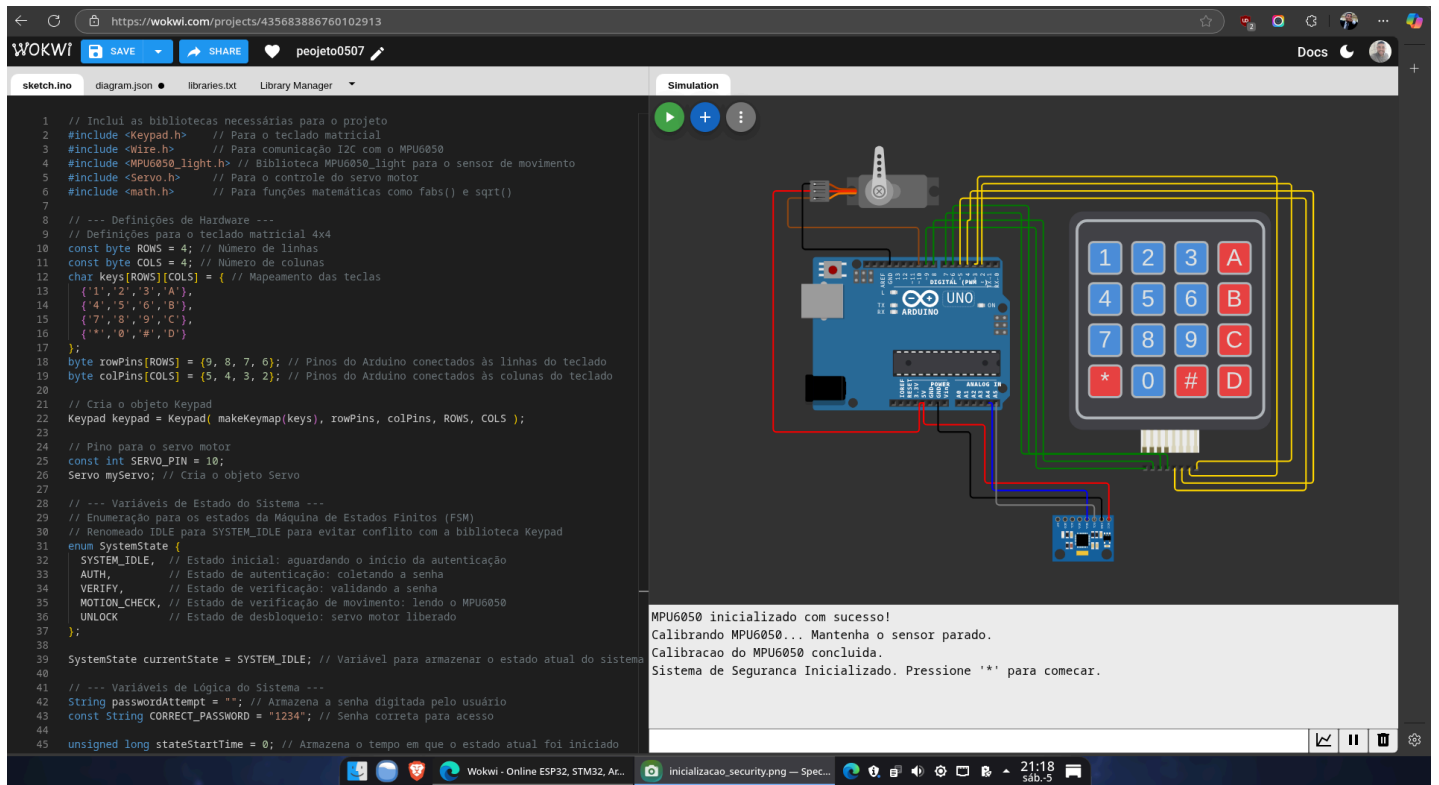
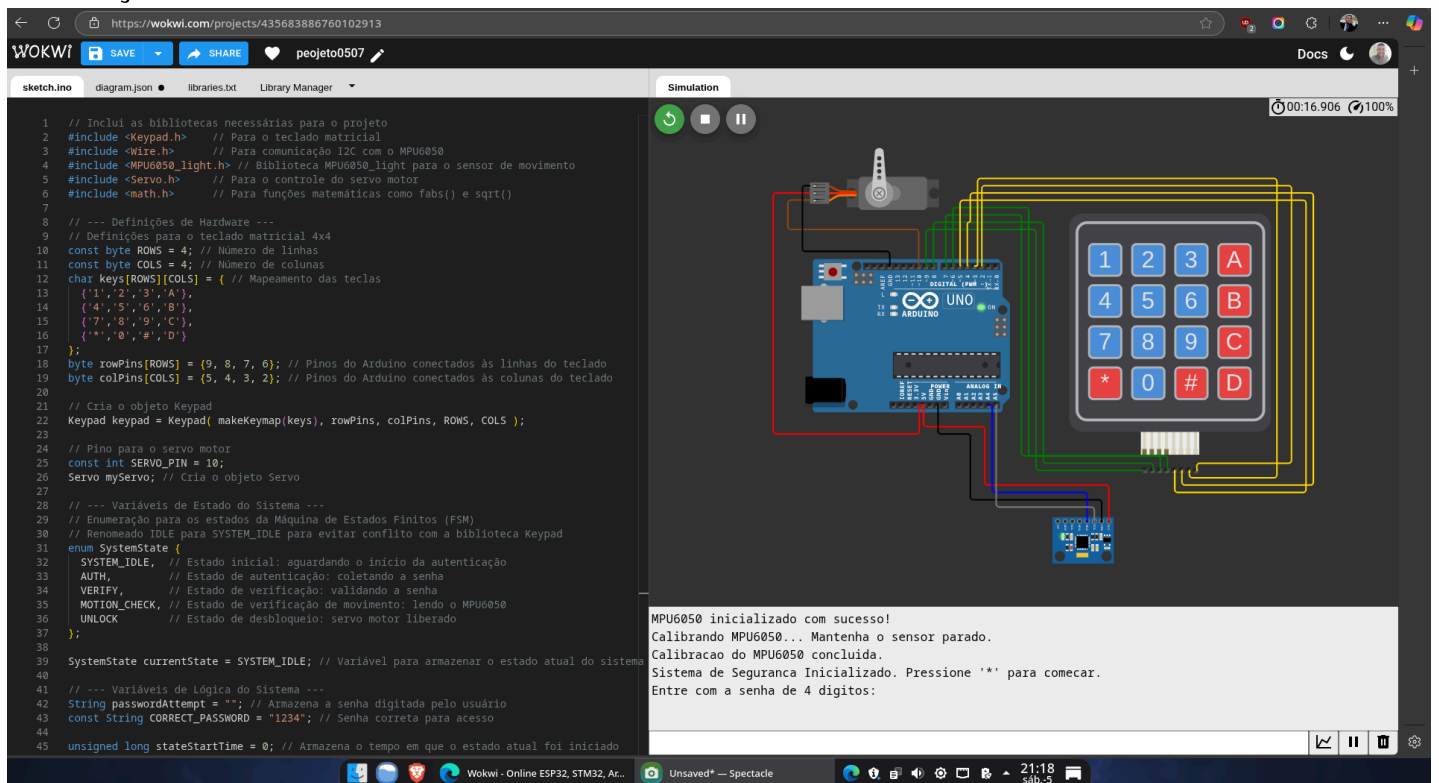
 **Nota:** Verifique se o endereço I2C da MPU é `0x68` , o padrão.

Imagem de Ligações:

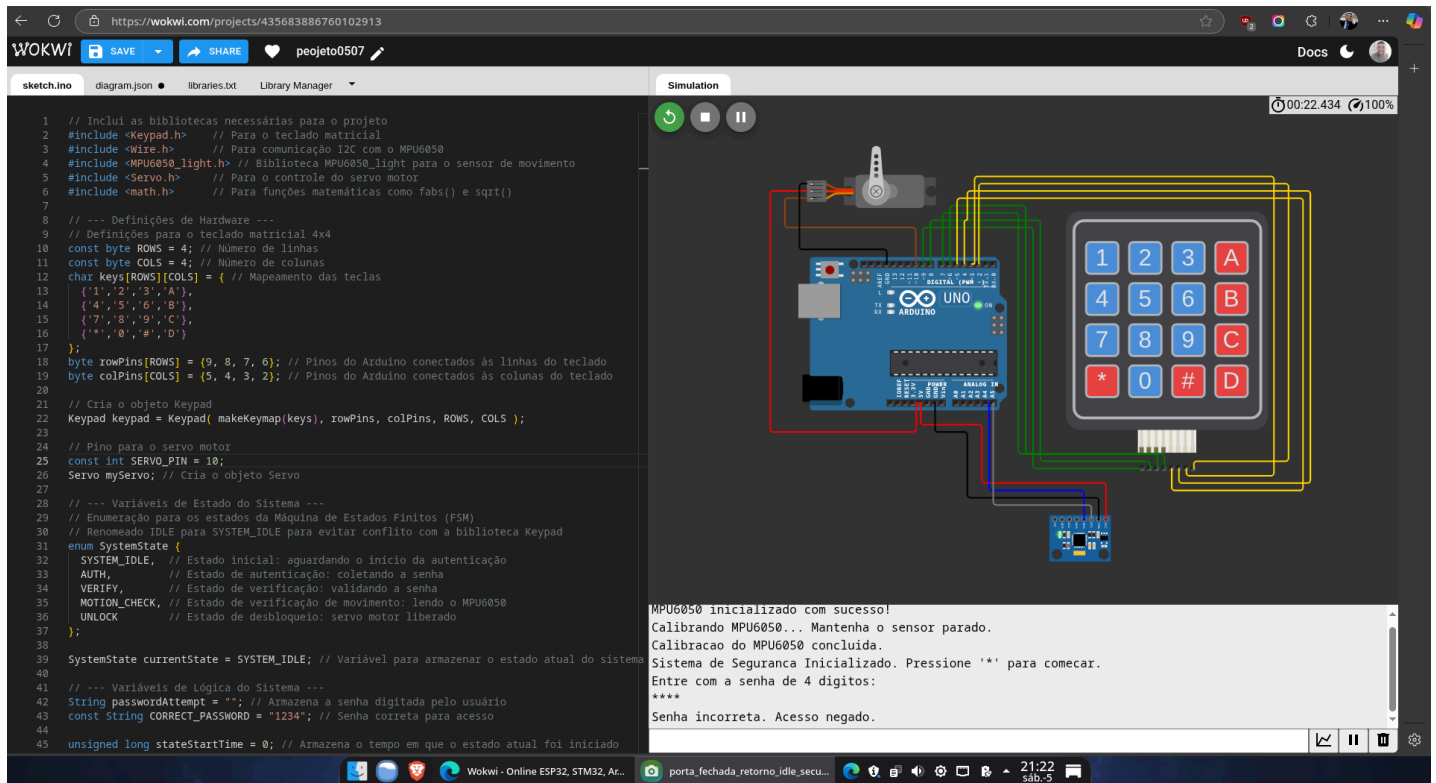
Sistema iniciado



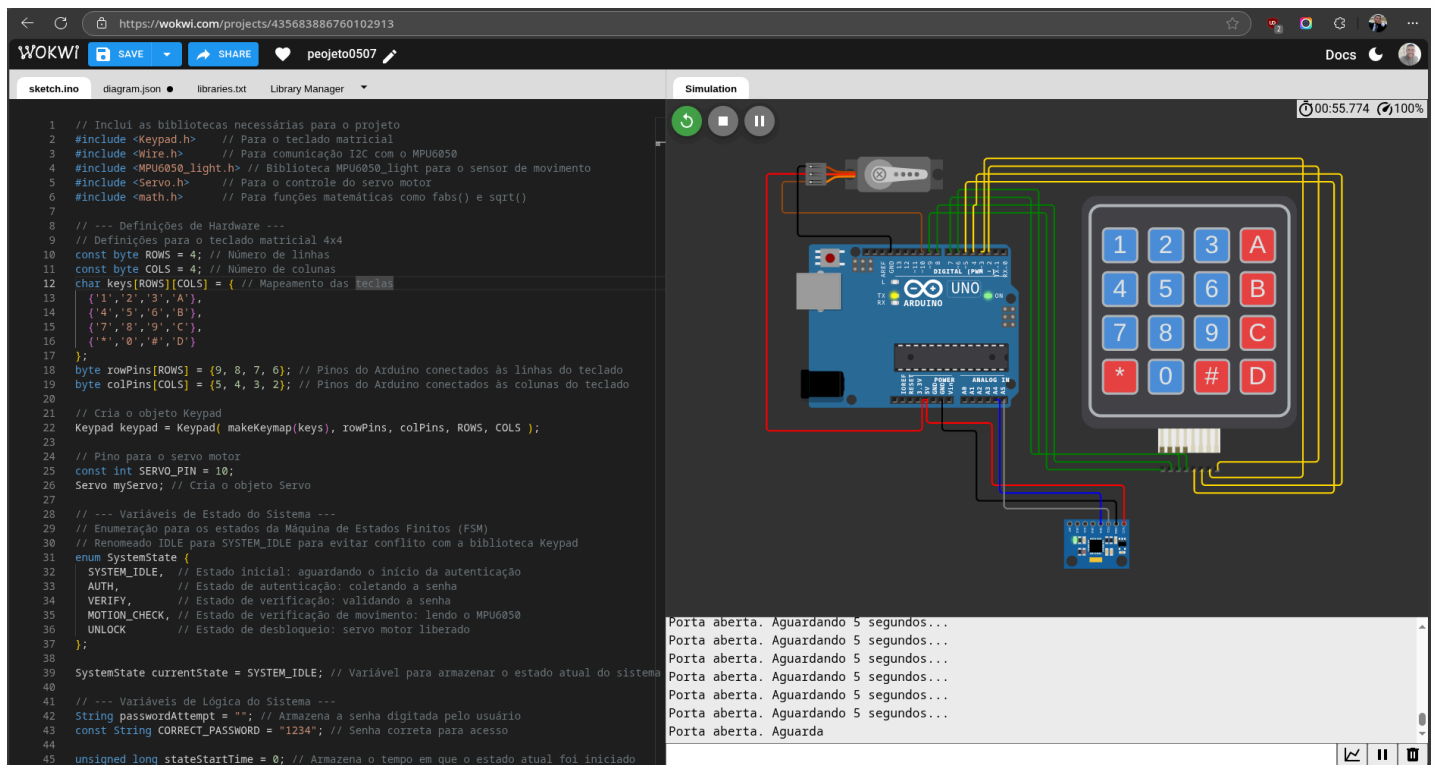
Solicitação de senha



Senha inserida e incorreta dá Acesso Negado



Senha inserida e correta Porta Abre



Após 5 segundos porta Fechada e Retorno ao IDLE

WOKWI

SAVE SHARE projeto0507 Docs

sketch.ino diagram.json libraries.txt Library Manager

```
1 // Inclui as bibliotecas necessárias para o projeto
2 #include <Keypad.h> // Para o teclado matricial
3 #include <Wire.h> // Para comunicação I2C com o MPU6050
4 #include <MPU6050_light.h> // Biblioteca MPU6050_light para o sensor de movimento
5 #include <Servo.h> // Para o controle do servo motor
6 #include <math.h> // Para funções matemáticas como fabs() e sqrt()
7
8 // --- Definições de Hardware ---
9 // Definições para o teclado matricial 4x4
10 const byte ROWS = 4; // Número de linhas
11 const byte COLS = 4; // Número de colunas
12 char keys[ROWS][COLS] = { // Mapeamento das teclas
13   {'1','2','3','A'},
14   {'4','5','6','B'},
15   {'7','8','9','C'},
16   {'*','0','#','D'}
17 };
18 byte rowPins[ROWS] = {9, 8, 7, 6}; // Pinos do Arduino conectados às linhas do teclado
19 byte colPins[COLS] = {5, 4, 3, 2}; // Pinos do Arduino conectados às colunas do teclado
20
21 // Cria o objeto Keypad
22 Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
23
24 // Pino para o servo motor
25 const int SERVO_PIN = 10;
26 Servo myServo; // Cria o objeto Servo
27
28 // --- Variáveis de Estado do Sistema ---
29 // Enumeração para os estados da Máquina de Estados Finitos (FSM)
30 // Renomeado IDLE para SYSTEM_IDLE para evitar conflito com a biblioteca Keypad
31 enum SystemState {
32   SYSTEM_IDLE, // Estado inicial: aguardando o início da autenticação
33   AUTH, // Estado de autenticação: coletando a senha
34   VERIFY, // Estado de verificação: validando a senha
35   MOTION_CHECK, // Estado de verificação de movimento: lendo o MPU6050
36   UNLOCK, // Estado de desbloqueio: servo motor liberado
37 };
38 SystemState currentState = SYSTEM_IDLE; // Variável para armazenar o estado atual do sistema
39
40 // --- Variáveis de Lógica do Sistema ---
41 String passwordAttempt = ""; // Armazena a senha digitada pelo usuário
42 const String CORRECT_PASSWORD = "1234"; // Senha correta para acesso
43
44 unsigned long stateStartTime = 0; // Armazena o tempo em que o estado atual foi iniciado
```

Simulation

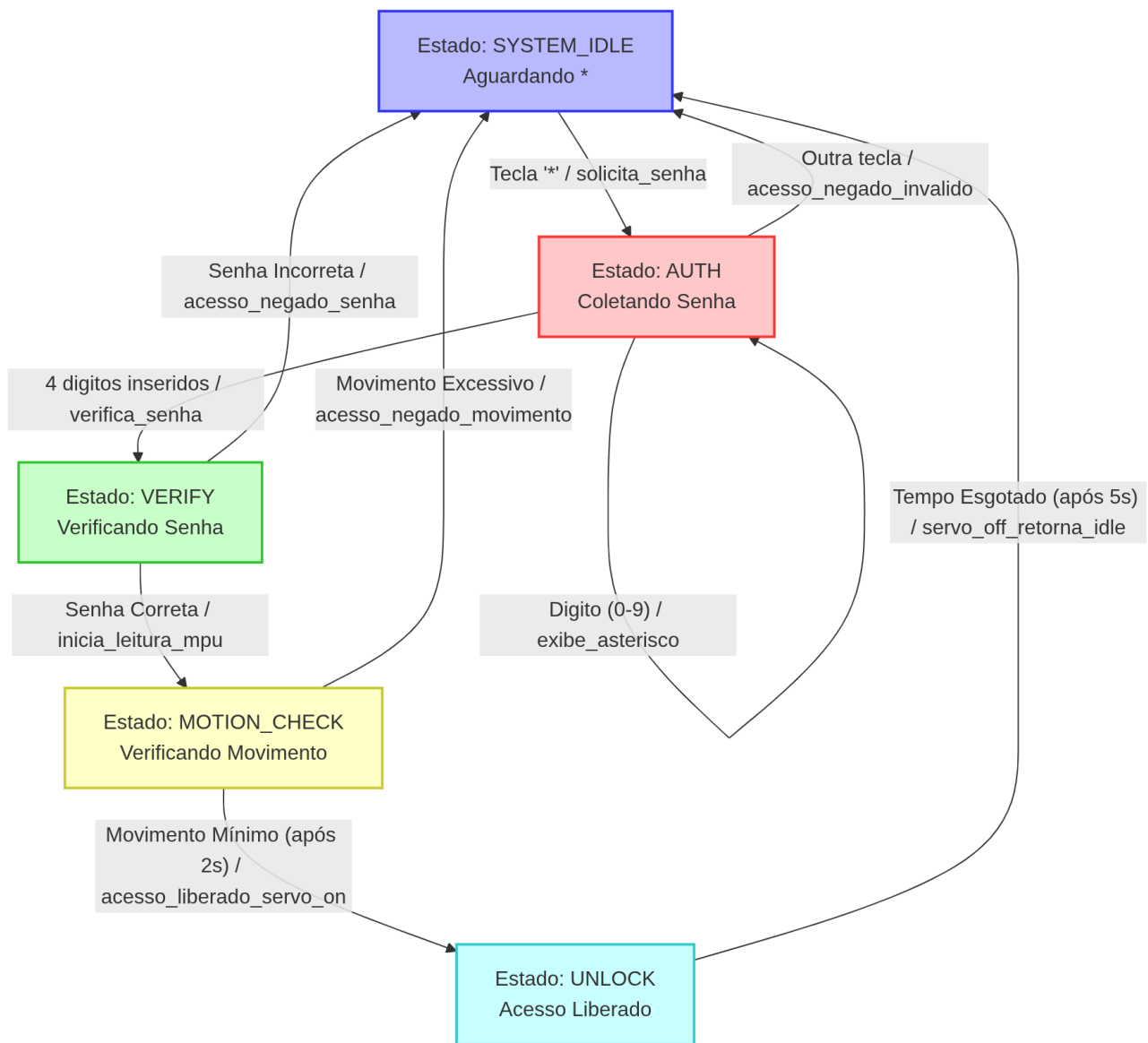
02:18.850 100%

Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta aberta. Aguardando 5 segundos...
Porta fechada. Sistema retornou ao Idle.

Diagrama da FSM de Mealy

Estados principais: - **SYSTEM_IDLE** : Espera o início (*) - **AUTH** : Coleta senha (4 dígitos) - **VERIFY** : Verifica senha (1234) - **MOTION_CHECK** : Aguarda imobilidade por 2s com MPU6050 - **UNLOCK** : Libera acesso por 5s com o servo

FSM em Imagem:



Código Arduino

Implementado em C++ com estrutura `switch-case` , utilizando:

- `Keypad.h` , `Wire.h` , `MPU6050_light.h` , `Servo.h`
- Funções auxiliares: `checkPassword()` e `checkMotion()`

Exemplo do Código Principal:

Acesse o código em [src/main_system_sketch.ino](#)

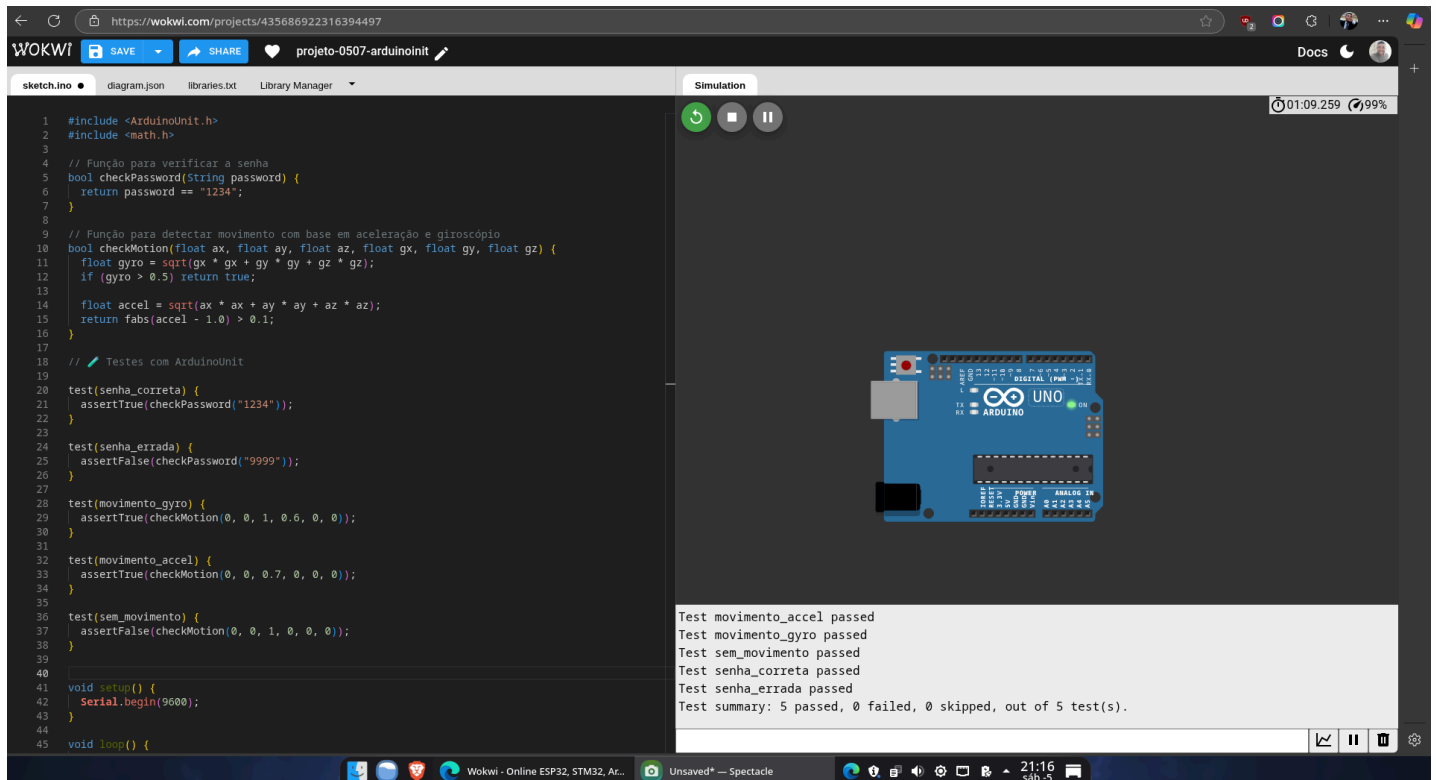
Testes com ArduinoUnit

Foram testadas as funções:

- `checkPassword()` — Verifica senha correta e incorreta
- `checkMotion()` — Detecta movimento via acelerômetro e giroscópio Utilizando o

código: [src/unit_tests_sketch.ino](#)

Screenshot do Serial Monitor:



Verificação Formal com CBMC

As funções críticas foram exportadas em C para análise com CBMC:

- `checkPassword.c`
- `checkMotion.c`

Verificações realizadas via terminal Linux com:

```
cbmc checkPassword.c --function main --unwind 1 --trace
cbmc checkMotion.c --function main --unwind 1 --trace
```

Screenshot do Terminal CBMC:

Tela de teste CBMC para Senhas

```

> cbmc checkPassword.c --function main --unwind 1 --trace
CBMC version 6.6.0 (6f6e8d8) 64-bit x86_64 linux
Type-checking checkPassword
file checkPassword.c line 22 function main: function '__CPROVER_havoc_memory' is not declared
Generating GOTO Program
Adding CPROVER library (x86_64)
Removal of function pointers and virtual functions
Generic Property Instrumentation
Starting Bounded Model Checking
Passing problem to propositional reduction
converting SSA
Running propositional reduction
SAT checker: instance is SATISFIABLE
Building error trace

** Results:
<builtin-library-strcmp> function strcmp
[strcmp.pointer_dereference.1] line 30 dereference failure: pointer NULL in s1[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.2] line 30 dereference failure: pointer invalid in s1[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.3] line 30 dereference failure: deallocated dynamic object in s1[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.4] line 30 dereference failure: dead object in s1[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.5] line 30 dereference failure: pointer outside object bounds in s1[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.6] line 30 dereference failure: invalid integer address in s1[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.7] line 31 dereference failure: pointer NULL in s2[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.8] line 31 dereference failure: pointer invalid in s2[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.9] line 31 dereference failure: deallocated dynamic object in s2[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.10] line 31 dereference failure: dead object in s2[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.11] line 31 dereference failure: pointer outside object bounds in s2[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.12] line 31 dereference failure: invalid integer address in s2[(signed long int)i]: SUCCESS

<builtin-library-strlen> function strlen
[strlen.pointer_dereference.1] line 18 dereference failure: pointer NULL in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.2] line 18 dereference failure: pointer invalid in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.3] line 18 dereference failure: deallocated dynamic object in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.4] line 18 dereference failure: dead object in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.5] line 18 dereference failure: pointer outside object bounds in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.6] line 18 dereference failure: invalid integer address in s[(signed long int)len]: SUCCESS
[strlen.unwind.0] line 18 unwinding assertion loop 0: FAILURE

checkPassword.c function main
[main.no-body.__CPROVER_havoc_memory] line 22 no body for callee __CPROVER_havoc_memory: FAILURE
[main.assertion.1] line 27 checkPassword deve retornar true para a senha correta: SUCCESS
[main.assertion.2] line 30 checkPassword deve retornar false para senha incorreta: SUCCESS

Trace for strlen.unwind.0:

```

Tela de teste CBMC para Motios (inconclusivo)

```

src: cbmc -- Console
[strcmp.pointer_dereference.10] line 31 dereference failure: dead object in s2[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.11] line 31 dereference failure: pointer outside object bounds in s2[(signed long int)i]: SUCCESS
[strcmp.pointer_dereference.12] line 31 dereference failure: invalid integer address in s2[(signed long int)i]: SUCCESS

<builtin-library-strlen> function strlen
[strlen.pointer_dereference.1] line 18 dereference failure: pointer NULL in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.2] line 18 dereference failure: pointer invalid in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.3] line 18 dereference failure: deallocated dynamic object in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.4] line 18 dereference failure: dead object in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.5] line 18 dereference failure: pointer outside object bounds in s[(signed long int)len]: SUCCESS
[strlen.pointer_dereference.6] line 18 dereference failure: invalid integer address in s[(signed long int)len]: SUCCESS
[strlen.unwind.0] line 18 unwinding assertion loop 0: SUCCESS

checkPassword.c function main
[main.no-body.__CPROVER_havoc_memory] line 23 no body for callee __CPROVER_havoc_memory: FAILURE
[main.assertion.1] line 31 checkPassword deve retornar true para a senha correta: SUCCESS
[main.assertion.2] line 34 checkPassword deve retornar false para senha incorreta: SUCCESS

** 1 of 22 failed (2 iterations)
VERIFICATION FAILED
[exit=10]
21:53:54
> cbmc checkMotion_cbm.c --bounds-check --pointer-check --unwind 1
CBMC version 6.6.0 (6f6e8d8) 64-bit x86_64 linux
Type-checking checkMotion_cbm
Generating GOTO Program
Adding CPROVER library (x86_64)
Removal of function pointers and virtual functions
Generic Property Instrumentation
Starting Bounded Model Checking
Passing problem to propositional reduction
converting SSA
Running propositional reduction
SAT checker: instance is SATISFIABLE
Running propositional reduction

70 | return 0; // indica sucesso na execução do programa (não na verificação)
71 | }
72 |

```

Interpretação dos Resultados do CBMC:

- **VERIFICATION SUCCESSFUL** : Significa que o CBMC não encontrou nenhum contra-exemplo que viole as asserções para o escopo explorado.
- **VERIFICATION FAILED** : Significa que o CBMC encontrou um contra-exemplo (um conjunto de valores de entrada que faz sua asserção falhar).



Material de Apoio e Referências

- **ArduinoUnit:** <https://github.com/mmurdoch/ArduinoUnit>
- **CBMC (C Bounded Model Checker):** <https://cprover.github.io/cbmc/>
- **MPU6050 com Arduino:** (Sugestão: procure por tutoriais da biblioteca `MPU6050_light` ou `Adafruit MPU6050`)
- **Servo Motor com Arduino:** (Sugestão: procure por tutoriais da biblioteca `Servo.h`)
- **Keypad com Arduino:** (Sugestão: procure por tutoriais da biblioteca `Keypad.h`)
- **Wokwi Arduino Simulator:** <https://wokwi.com/>

Conclusão

O sistema integra autenticação segura e verificação de movimento com testes formais e simulação. Toda a lógica segue o modelo FSM de Mealy e é validada por testes e verificação estática.