FH Joanneum

Graz

Model Based Design

Numeric Sequence Lock

Training Unit 04

Authors
David B. Heer
Jakob Soukup
Graz, November 27, 2018

Lecturer
Alfred Steinhuber

Contents

Introduction

The task was to implement an Finit State Machine in Simulink. Afterwards this Finit State Machine is to be tested for its requirements. Finally code should be generated and loaded onto the Arduino board.

1 Requirements

1.1 Order of Priorities

1. Unlocking
   This is the core of the task and so the most important requirement.

2. Locking
   It is necessary to return to the locked state.

3. Sampling
   With a sampletime of 10ms you do not have to debounce the inputs. In this case is sampling important.

4. Wrong Sequence
   This requirement makes it difficult to crack the code using the brute force method.

5. Input Handling
   This is a nice feature and can be useful if you have mistyped.

6. State
   It would be a nice feature to see the current state. But it is not essential.

7. Keypad
   It is absolutely irrelevant whether the numbers are compared as integers between 0-9 or as ACII integers between 48 - 57.

The Voltage Monitoring is an own requirement. Because it has no connection with the remaining requirements.

1.2 Funcional and non functional requirements

Functional

- Unlockin

- Wrong Sequence

- Input Handling

- State

- Sampling

- Locking

- Keypad

- Voltage Monitoring

Non functional

- sdgs

1.3   Missing requirements

Change Sequence   It would be useful if you could change the sequence while the program is running and not just when the program is freshly loaded onto the arduino board.

label   description

2   Stateflow implementation

2.1   Finit State Machine

2.2   Voltage Monitoring

3   Testing

4   Linkage/ Traceabity of requirements

4.1   Links between Models and Requirements

4.2   Create a Link from a Model Object to a Microsoft Word Requirements Document

4.3   Requirements visibility and navigation

4.4   Linkage of requirements to Signal Builder Block (Test Cases)

4.5   Traceability/Report

4.6   Automatic Test Case Generation for increasing coverage

5   Conclusion

List of Figures