Module Interface Specification for 4TB6 - Mechatronics Capstone

Team #5, Locked & Loaded
Abi Nevo, nevoa
Elsa Bassi, bassie
Steffi Ralph, ralphs1
Abdul Iqbal, iqbala18
Stephen De Jong, dejons1
Anthony Shenouda, shenoa2

January 17, 2023

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation here.

Contents

3 Introduction

The following document details the Module Interface Specifications for SmartLock, a bluetooth-driven bike lock brough to you by the Locked & Loaded team. The SmartLock allows users to unlock their bike remotely using Bluetooth.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found in the GitHub repo.

Note that not every module documented in the Module Guide has a corresponding section in this document, as an MIS was only completed for every software module, and not those modules with a hardware implementation.

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by 4TB6 - Mechatronics Capstone.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of 4TB6 - Mechatronics Capstone uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, 4TB6 - Mechatronics Capstone uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Input Parameters Module Output Parameters Module Engage Status Signal Module Disengage Signal Module Battery Status Module Location Module Lock Frame Module
Software Decision Module	Load Power Signal Module Battery Module Electromagnet Module Locking Mechanism Module

Table 1: Module Hierarchy

6 MIS of [Module Name —SS]

[Use labels for cross-referencing —SS]
[You can reference SRS labels, such as R??. —SS]
[It is also possible to use LaTeXfor hypperlinks to external documents. —SS]

6.1 Module

[Short name for the module —SS]

6.2 Uses

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

6.4 Semantics

6.4.1 State Variables

[Not all modules will have state variables. State variables give the module a memory. —SS]

6.4.2 Environment Variables

[This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

6.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

6.4.4 Access Routine Semantics

[accessProg —SS]():

- inputs:
- transition: [if appropriate —SS]

- output: [if appropriate —SS]
- exception: [if appropriate —SS]

6.4.5 Local Functions

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

7 MIS of M4

7.1 Engage Status Signal Module

As described in the MG document, the Engage Status Signal Module (M4) is responsible for trasmitting the engagement status of the lock from the Arduino to the mobile app. If the status reads "engaged", then the Arduino is not currently sending a high signal to the transistor, and the electromagnet remains off, meaning the latch in the locking mechanism is shut. Therefore, if the pin is in the lock, it will not be able to move. If the status reads "disengaged", then the Arduino is currently writing a high signal to the transistor, and the electromagnet is on, opening the latch in the locking mechanism, and allowing the pin to move freely (in or out of the lock).

7.2 Uses

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

N/A

7.4 Semantics

7.4.1 State Variables

None.

7.4.2 Environment Variables

- e_BTService, type: BLEService
- e_DisengageCharacteristic; type: BLEByteCharacteristic

Note that the environment variables are the same as that of M5, as an established Bluetooth connection is a prerequisite of M4, and these variables must still be used to keep the BlueTooth connection active.

7.4.3 Assumptions

This module assumes there is a successful BlueTooth connection established between the Arduino and the mobile app.

7.4.4 Access Routine Semantics

readEngagementStatus():

This access routine will be implemented on the mobile app, and will read the current value of e_DisengageCharacteristic.

- inputs
- transition:
- output: e_DisengageCharacteristic
- exception:

7.4.5 Local Functions

None.

8 MIS of M5

8.1 Wireless Signal Connection Module

As described in the MG document, the Wireless Signal Connection Module (M5) is responsible for establishing a BlueTooth connection between the Arduino and the mobile app.

8.2 Uses

8.3 Syntax

8.3.1 Exported Constants

N/A

8.3.2 Exported Access Programs

N/A

8.4 Semantics

8.4.1 State Variables

None.

8.4.2 Environment Variables

- e_BTService, type: BLEService
- e_DisengageCharacteristic; type: BLEByteCharacteristic

8.4.3 Assumptions

- Arduino is powered on.

8.4.4 Access Routine Semantics

BTconnect():

This routine creates a BlueTooth connection between the mobile app and the Arduino so that they can send and receive signals from each other. This routine will need an implementation both for the Arduino, and for the mobile app, where the Arduino will act as the peripheral device, and the mobile app will act as the central device.

- inputs: none
- transition: loadPower(), the access routine of M7, upon successful connection
- output: none
- exception: connection status will appear on the mobile app. Therefore, the user will be aware of the connection status, whether that be successful or unsuccessful.

8.4.5 Local Functions

None.

9 MIS of M7

9.1 Load Power Signal Module

As described in the MG document, the Load Power Signal Module (M7) is responsible for sending a high power/ON signal to the transistor once a disengage signal is written to the Arduino. An ON signal to the transistor acts as a switch ON, and will power the electromagnet to disengage the lock.

9.2 Uses

9.3 Syntax

9.3.1 Exported Constants

N/A

9.3.2 Exported Access Programs

N/A

9.4 Semantics

9.4.1 State Variables

None.

9.4.2 Environment Variables

- e_BTService, type: BLEService
- e_DisengageCharacteristic; type: BLEByteCharacteristic

Note that the environment variables are the same as that of M5, as an established Bluetooth connection is a prerequisite of M7, and these variables must still be used to keep the BlueTooth connection active.

9.4.3 Assumptions

Assumes M5 has been successfully completed; there is an established BlueTooth connection between the Arduino and the mobile app.

9.4.4 Access Routine Semantics

loadPower():

This access routine is responsible for recieving the signal to disengage the lock, and then, should this signal be received, sending a HIGH signal to the transistor. This will be implemented on the Arduino.

- inputs: e_DisengageCharacteristic
- transition: none
- output: if e_DisengageCharacteristic has a nonzero value (i.e., the disengage button on the app GUI is pressed), write a HIGH signal to the Arduino pin wired to the corresponding transistor terminal for five seconds (enough time to pull the pin out of the lock, or in other words, unlock your bike).
- exception: none

9.4.5 Local Functions

None.

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

10 Appendix

[Extra information if required —SS]