Module Interface Specification for SmartLock 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

April 7, 2023

Table 1: Revision History

Date	**Developer(s)**	**Change**
16-01-23	Abi	Started M4, M5, M7
17-01-23	Abi	Finished M4, M5, M7, Intro
18-01-23	Anthony	Finished M2,3,6,8
07-04-23	Abi and Anthony	Rev 1 updates

Contents

1 Symbols, Abbreviations and Acronyms **1**

2 Introduction **1**

3 Notation **1**

4 Module Decomposition **1**

5 MIS of Arduino Bluetooth Communication Module **2**
5.1 Module . . . 2
5.2 Syntax . . . 2
5.2.1 Exported Constants . . . 2
5.2.2 Exported Access Programs . . . 2
5.3 Semantics . . . 2
5.3.1 State Variables . . . 2
5.3.2 Access Routine Semantics . . . 2
5.3.3 Local Functions . . . 3

6 MIS of Hardware Disengage Module **3**
6.1 Module . . . 3
6.2 Syntax . . . 3
6.2.1 Exported Constants . . . 3
6.2.2 Exported Access Programs . . . 4
6.3 Semantics . . . 4
6.3.1 State Variables . . . 4
6.3.2 Access Routine Semantics . . . 4
6.3.3 Local Functions . . . 5

7 MIS of Mobile App Bluetooth Communication and User Disengage Module **5**
7.1 Module . . . 5
7.2 Semantics . . . 5
7.2.1 State Variables . . . 5
7.2.2 Local Functions . . . 5

8 MIS of Battery Status Module **6**
8.1 Module . . . 6
8.2 Semantics . . . 6
8.2.1 State Variables . . . 6
8.2.2 Local Functions . . . 6

9 MIS of Location Module **7**
9.1 Module 7
9.2 Semantics 7
9.2.1 State Variables 7
9.2.2 Local Functions 8

List of Tables

1 Revision History i
2 Module Hierarchy 1

1 Symbols, Abbreviations and Acronyms

See SRS Documentation here.

2 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.

3 Notation

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.

4 Module Decomposition

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

Level 1	**Level 2**
Hardware-Hiding Module	User Input to Phone Module Solenoid Actuation Module
Behaviour-Hiding Module	Arduino Bluetooth Communication Module Mobile App Bluetooth Communication Module Battery Status Module Location Module Lock Frame Module
Software Decision Module	User Disengage Module Hardware Disengage Module Battery Module Locking Mechanism Module

Table 2: Module Hierarchy

5 MIS of Arduino Bluetooth Communication Module

5.1 Module

disengageControl.ino

5.2 Syntax

5.2.1 Exported Constants

Name	Value	Description
BAUD_RATE	9600	Serial communication baud rate
GREEN_PIN	23	Pin for green LED onboard

5.2.2 Exported Access Programs

Name	In	Transition	Out	Exception
setup	None	loop	None	BLE fails: print message
loop	None	Hardware Disengage Module	None	BLE fails: print message

5.3 Semantics

5.3.1 State Variables

Name	Type	Description
nanoService	BLEService	BLE service UUID
commCharacteristic	BLEByteCharacteristic	BLE Characteristic UUID
password	int	password for disengage
productName	char array	name to appear when device BL advertised
central	BLEDevice	connected central device

5.3.2 Access Routine Semantics

setup

- Begins serial communication operating at BAUD_RATE
- Configures output pins specified by Exported Constants
- Configures BLE settings: name, advertising service, characteristic, and initial value for the characteristic. Do this using functions from the Arduino Bluetooth library, use *#include ArduinoBLE.h*
- Begins advertising BL service, and awaits for connections, prints status message

loop

- Listens for central device to connect using aforementioned Bluetooth library
- If a central device is connected, print a statement of this status and transition to Hardware Disengage Module
- When the central device disconnects, print a message indicating this new status

5.3.3 Local Functions

A list of functions used in this module from Arduino Bluetooth library, see official Arduino documentation here

- setDeviceName()
- setLocalName()
- setAdvertisedService()
- addCharacteristic()
- addService()
- writeValue()
- advertise()

6 MIS of Hardware Disengage Module

6.1 Module

disengageControl.ino

6.2 Syntax

6.2.1 Exported Constants

Name	Value	Description
BAUD_RATE	9600	Serial communication baud rate
GREEN_PIN	23	Pin for green LED onboard
TRANSISTOR_OUT	9	Output pin for signal to transistor

6.2.2 Exported Access Programs

Name	In	Transition	Out	Exception
setup	None	loop	None	-
loop	None	see Access Routine Semantics	None	-

6.3 Semantics

6.3.1 State Variables

Name	Type	Description
password	int	password for disengage
central	BLEDevice	connected central device
commCharacteristic	BLEByteCharacteristic	BLE Characteristic UUID

6.3.2 Access Routine Semantics

setup

- Begins serial communication operating at BAUD_RATE
- Configures output pins specified by Exported Constants

loop

- While a central device is connected;
- If the central device writes a value to the Arduino;
- If the written value matches *password*;
- Then print a message indicating this status, turn on the onboard green LED (write a LOW signal to GREEN_PIN, and write a HIGH signal to TRANSISTOR_OUT;
- Else, turn off the onboard green LED (write a HIGH signal to GREEN_PIN, and write a LOW signal to TRANSISTOR_OUT;
- This logic can also be described formally, using discrete math, with the following:

  If the central device is connected:

  $$central = C = \text{True}$$

  If the central device writes a value to the Arduino:

  $$commCharacteristic.written() = W = \text{True}$$

If the written value matches *password*:

$$P = \text{True}$$

If P is true, then print a message indicating the status, turn on the onboard green LED, and set the transistor output high:

$$P \Rightarrow (GREEN\_PIN = \text{LOW} \wedge TRANSISTOR\_OUT = \text{HIGH})$$

If P is false, then turn off the onboard green LED, and set the transistor output low:

$$P \Rightarrow (GREEN\_PIN = \text{HIGH} \wedge TRANSISTOR\_OUT = \text{LOW})$$

6.3.3 Local Functions

No local functions.

7 MIS of Mobile App Bluetooth Communication and User Disengage Module

7.1 Module

device.dart

7.2 Semantics

7.2.1 State Variables

Name	Type	Description
locking	int	request to engage the lock (value of 1 to engage)
isConnected	int	whether SmartLock App is connected to the Arduino (value of 1 if connected)
password	int	the value of the user inputted password

7.2.2 Local Functions

buildServiceTiles

- Constructs the layout of the screen displaying the connected device
- Configures the buttons to disengage lock by sending the user inputted password to the Arduino

- inputs: password
- transition: none
- output: password to Arduino
- exception: none

Widget build (BuildContext context)

- Constructs the layout of the screen displaying the bluetooth devices
- Displays the connection status of the selected bluetooth device
- inputs: none
- transition: none
- output: connection status displayed to user and device information
- exception: none

8 MIS of Battery Status Module

8.1 Module

main.dart

8.2 Semantics

8.2.1 State Variables

Name	Type	Description
batteryPercentage	int	the calculated value of the battery percentage
completedActuations	int	the total number of actuations to engage the lock completed

8.2.2 Local Functions

getLocalPath

- This functions gets the local directory of the application

getLocalFile

- This function gets the local file that stores the battery percentage and number of completed actuations

- It searches for a file called 'batteryInfo.txt'

writeBattery

- This function reads the value of the calculated battery percentage from the 'batteryInfo.txt' file

readBattery

- This function writes the value of the calculated battery percentage to the 'batteryInfo.txt' file
- If this file can not be found (if this the first time the application is run) it will create a new file titled 'batteryInfo.txt'

batteryCalculator

- This function is used to calculate the amount of battery remaining
- This is done by dividing the total capacity of the battery by the power drawn from completing on actuation of engaging the lock:
  Battery Life $= \frac{\text{Total Capacity of Battery}}{\text{Power Drawn from Completing One Actuation of Engaging the Lock}}$
- inputs: batteryPercentage
- transition: none
- output: new battery percentage
- exception: none

9 MIS of Location Module

9.1 Module

currentLocation.dart

9.2 Semantics

9.2.1 State Variables

Name	Type	Description
locLAT	double	the latitude coordinates of the users current location
locLOG	double	the longitude coordinates of the users current location

9.2.2 Local Functions

getLocalPath

- This functions gets the local directory of the application

getLocalFileLat

- This function gets the local file that stores the latitude coordinates
- It searches for a file called 'locationLat.txt'

getLocalFileLog

- This function gets the local file that stores the longitude coordinates
- It searches for a file called 'locationLog.txt'

readLat

- This function reads the value of the latitude coordinates from the 'locationLat.txt' file
- If this file can not be found (if this the first time the application is run) it will create a new file titled 'locationLat.txt'

readLog

- This function reads the value of the longitude coordinates from the 'locationLog.txt' file
- If this file can not be found (if this the first time the application is run) it will create a new file titled 'locationLog.txt'

writeLat

- This function writes the value of the latitude coordinates to the 'locationLat.txt' file

writeLog

- This function writes the value of the latitude coordinates to the 'locationLog.txt file

Widget build(BuildContext context)

- This function builds the layout of the screen including the Google Maps API and getCurrentLocation button
- When interacted with, it places a marker showing the current location of the user and stores those coordinates
- inputs: locationLat and locationLog

- transition: none
- output: new locationLat and locationLog
- exception: none

determinePosition()

- This function gets user permission to use their mobile GPS
- inputs: none
- transition: none
- output: none
- exception: If the user denies permission then disable Google Maps API