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

April 5, 2023

1 Revision History

Date	Version	Notes
16/01/23	1.0	Abi started M4, M5, M7
17/01/23	1.1	Abi finished M4, M5, M7, Intro
18/01/23	1.2	Anthony finished M2,3,6,8

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

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	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 1: Module Hierarchy

6 MIS of Arduino Bluetooth Communication 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

6.2.2 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

6.3 Semantics

6.3.1 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

6.3.2 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

6.3.3 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()

7 MIS of Hardware Disengage Module

7.1 Module

disengageControl.ino

7.2 Syntax

7.2.1 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

7.2.2 Exported Access Programs

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

7.3 Semantics

7.3.1 State Variables

Name	Type	Description
password	int	password for disengage
central	BLEDevice	connected central device

7.3.2 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;

7.3.3 Local Functions

No local functions.

8 MIS of M3

8.1 Output Parameter Module

As described in the MG document, the Output Parameter Module (M3) is responsible for displaying the battery status, location, and the engaged status signal. In this module, there are submodules to store the current location requested by the user as coordinates on a text file.

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

- locationLatitude, type: String
- locationLongitude, type: String

8.4.2 Environment Variables

None.

8.4.3 Assumptions

None.

8.4.4 Access Routine Semantics

loadScreen(): This routine oversees the layout and UserInterface architecture. It ensures all buttons, labels, and images fall within the constraints of any mobile device.

- inputs: storedCurrentLocationButton, bluetoothConnectButton, getLocationButton, batteryButton
- transition: bluetoothConnectButton, getLocationButton
- output: display current location and battery percentage
- exception: none

8.4.5 Local Functions

None.

9 MIS of M4

9.1 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).

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 M4, and these variables must still be used to keep the BlueTooth connection active.

9.4.3 Assumptions

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

9.4.4 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:

9.4.5 Local Functions

None.

10 MIS of M5

10.1 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.

10.2 Uses

10.3 Syntax

10.3.1 Exported Constants

N/A

10.3.2 Exported Access Programs

N/A

10.4 Semantics

10.4.1 State Variables

None.

10.4.2 Environment Variables

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

10.4.3 Assumptions

- Arduino is powered on.

10.4.4 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.

10.4.5 Local Functions

None.

11 MIS of M6

11.1 Battery Status Module

As described in the MG document, the Battery Status Module (M6) is responsible for calculating the battery status. In this module, there are submodules to calculate the amount of battery left. The percentage is shown as a percent of the number of actuations that can be completed till the battery loses power.

11.2 Uses

11.3 Syntax

11.3.1 Exported Constants

N/A

11.3.2 Exported Access Programs

N/A

11.4 Semantics

11.4.1 State Variables

None.

11.4.2 Environment Variables

None.

11.4.3 Assumptions

There is an assumption that the power drawn from the Arduino is negligible. Therefore it is not taken into consideration when calculating the battery percentage.

11.4.4 Access Routine Semantics

getBatteryStatus(): This routine gets the battery amount remaining. This is displayed to the user as a percentage of the number of actuations remaining.

- inputs: batteryCalculator()
- transition: none
- output: battery percentage as displayed to the user
- exception: none

11.4.5 Local Functions

batteryCalculator(): This function is used to calculate the amount of battery remaining. The output is used in the getBatteryStatus function. The function works by reading the battery percentage from the local text file. It then calculated the remaining battery life by taking how much of the battery has been used and calculating how much is remaining. It then writes that amount back to the local text file.

12 MIS of M7

12.1 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.

12.2 Uses

12.3 Syntax

12.3.1 Exported Constants

N/A

12.3.2 Exported Access Programs

N/A

12.4 Semantics

12.4.1 State Variables

None.

12.4.2 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.

12.4.3 Assumptions

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

12.4.4 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

12.4.5 Local Functions

- writeBattery()
- getBattery()

13 MIS of M8

13.1 Location Module

As described in the MG document, the Location Module (M8) is responsible for gathering the location data requested by the user. In this module, there are submodules to store the current location requested by the user. The module utilizes the Google Maps API to get the current user location and display it on the main screen.

13.2 Uses

13.3 Syntax

13.3.1 Exported Constants

N/A

13.3.2 Exported Access Programs

N/A

13.4 Semantics

13.4.1 State Variables

- locationLatitude, type: String
- locationLongitude, type: String

13.4.2 Environment Variables

None.

13.4.3 Assumptions

The user is storing their location right next to where they are leaving their bike.

13.4.4 Access Routine Semantics

updateLocation(): This routine writes to the local text file to store the last location requested by the user.

- inputs: locationPinButton()
- transition: none
- output: none
- exception: none

getLocation(): This routine reads from the local text file to get the last location requested by the user

- inputs: none
- transition: none
- output: storedCurrentLocationButton
- exception: none

13.4.5 Local Functions

- getLatLocation()
- getLogLocation()
- writeLatLocation()
- writeLogLocation()