Module Interface Specification for SE 4G06, TRON 4TB6

Team 26, STRONE
Jordan Bierbrier
Azriel Gingoyon
Taranjit Lotey
Udeep Shah
Abraham Taha

January 18, 2023

Revision History

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

Symbols, Abbreviations and Acronyms

See SRS Documentation at *SRS.pdf Document Link*.

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

1 Introduction

The following document details the Module Interface Specifications for Synesthesia Wear, a wearable product that assists users by using signal processing on gathered sounds to provide appropriate feedback (via vibrations) to the user according to inputted sound configuration settings. As a result, this gives the users peace of mind knowing that if their attention is needed (doorbell, ring, name call, etc.), Synesthesia Wear will be able to alert them.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at *Team 26 Capstone GitHub Repository*.

2 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 **?**, with the addition that template modules have been adapted from **?**. The mathematical notation comes from Chapter 3 of **?**. 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 SE 4G06, TRON 4TB6.

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 SE 4G06, TRON 4TB6 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, SE 4G06, TRON 4TB6 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.

3 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	Login Module Bluetooth connection Module Keyword Selection Module Output Signal Module Profile Module Battery Status Module
Software Decision Module	Sound Classification Module Bluetooth Communication Module Microphone Module

Table 1: Module Hierarchy

4 MIS of Battery Status

4.1 Module

Battery Status

4.2 Uses

Indicator

4.3 Syntax

4.3.1 Exported Constants

N/A

4.3.2 Exported Access Programs

Name	In	Out	Exceptions
Low Battery	N/A	Low Status	Low Battery
Battery Level	N/A	Percentage	Battery Status

4.4 Semantics

4.4.1 State Variables

N/A

4.4.2 Environment Variables

keyOutput : key.Low Battery, key.Battery Status

4.4.3 Assumptions

Battery will always have some charge on it.

4.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: N/A
- output: Battery level
- exception: N/A

4.4.5 Local Functions

- get level(int level)
- battery condition(int cond)

5 MIS of Battery Status

5.1 Module

Sound Classification

5.2 Uses

Categorize detected sounds

5.3 Syntax

5.3.1 Exported Constants

N/A

5.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
Sound level	Microphone	decibels	High/Low Level

5.4 Semantics

5.4.1 State Variables

N/A

5.4.2 Environment Variables

keyOutput : key.microphone keyOutput : key.decibels

5.4.3 Assumptions

Surrounding sound levels are null compared to required input level.

5.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: N/A
- output: decibels
- exception: N/A

5.4.5 Local Functions

- sound level(float volume)
- sorting volume()

6 MIS of Login Module

6.1 Module

Login Type

6.2 Uses

N/A

6.3 Syntax

6.3.1 Exported Constants

N/A

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
authorize	keyInput	Authorized	
login	keyInput		Not_A_Character
switchCue	keyInput		Not_Cueable

6.4 Semantics

6.4.1 State Variables

Authorized: Authorized is a boolean that is true when the correct credentials are entered and false otherwise.

6.4.2 Environment Variables

keyInput: {key.Enter, key.AlphabetCharacters, key.LeftClick}

6.4.3 Assumptions

The Synesthesia Wear application is successfully installed on the user's device and the login page has loaded onto the screen.

6.4.4 Access Routine Semantics

switchCue(key.LeftClick):

- transition: mouseLocation.navigate()
- output: None
- exception: Not_Cueable

login(key.AlphabetCharacters):

- transition: username.addCharacter() or password.addCharacter()
- output: None
- exception: Not_A_Character

authorize(key.Enter):

- transition: login.submit()
- output: Authorized
- exception: None

7 MIS of Keyword Selection Module

7.1 Module

Keyword Selection Type

7.2 Uses

- Profile Module
- Bluetooth Communication Module

7.3 Syntax

7.3.1 Exported Constants

N/A

7.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
save	keyInput	Saved	Mouse_Not_On_Save_Button
keyword	keyInput		Not_A_Character
switchCue	keyInput		Not_Cueable

7.4 Semantics

7.4.1 State Variables

Saved: Saved is a boolean that is true when the "Save" button has been pressed and false otherwise.

7.4.2 Environment Variables

keyInput: {key.AlphabetCharacters, key.LeftClick}

7.4.3 Assumptions

The Synesthesia Wear application is successfully installed on the user's device, the user was able to log into the app, and the sound configuration settings page is loaded onto the screen.

7.4.4 Access Routine Semantics

switchCue(key.LeftClick):

- transition: mouseLocation.navigate()
- output: None
- exception: Not_Cueable

keyword(key.AlphabetCharacters):

- transition: keyword.addCharacter()
- output: None
- exception: Not_A_Character

save(key.LeftClick):

- transition: keyword.save()
- output: Saved
- exception: Mouse_Not_On_Save_Button

8 MIS of Bluetooth Communication Module

8.1 Module

BTComuModule

8.2 Uses

None

8.3 Syntax

8.3.1 Exported Constants

DataTx:{ BT.send()}

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
RecvNewClass	DataRx	-	Class_Full
RmClass	DataRx	-	Class_Empty
UpdatePref	DataRx	-	-
SendBattV	BattV	DataTx	-

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Environment Variables

BattV: {AnalogRead(BatteryVolt)}
DataRx: {BT.recieve(T1),BT.recieve(T2),BT.recieve(T3)}

8.4.3 Assumptions

Connection with the application is already establised. Bluetooth tries to automatically reconnect if the application is disconnected.

8.4.4 Access Routine Semantics

RecvNewClass():

- transition: if(DataRx == T1) then addclass(dataRx)

- output: None
- exception: Class_Full

RmClass():

- transition: if(DataRx == T2) then rmclass(dataRx)
- output: None
- exception: Class_Empty

UpdatePref():

- transition: if(DataRx == T3) then prefchange(DataRx.class,DataRx.pref)
- output: None
- exception: None

SendBattV(BattV):

- transition: None
- output: DataTx == BattV
- exception:

9 MIS of Microphone Module

9.1 Module

MicroMod

9.2 Uses

None

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
updateData	SoundIN,CLKINT	SoundData	-

9.4 Semantics

9.4.1 State Variables

SoundData - Array of sound inputs recorded at every sample interval

9.4.2 Environment Variables

SoundIN: { Digital.Read(Digital_Microphone) }
CLKINT: {clk.interupt(sample time)}

9.4.3 Assumptions

Clock interupt is already set up to issue interupts every 1/(sample frequency)

9.4.4 Access Routine Semantics

updateData(SoundIN,CLKINT):

- transition: None
- output: if(CLKINT) then (SoundData = SoundData ¿¿ 1, SoundData[new] = SoundIN)
- exception: None

10 Appendix

[Extra information if required —SS]