Software Simulator: Microcurrent Biofeedback Device

Background.

The purpose of the deliverable is to break down all required components, use cases and present an object-oriented design for a non-invasive medical device that delivers modulated electrical signals via an electrode through the skin to communicate with the peripheral nervous system for the purpose of therapeutic intervention. The device will be based on the DENAS and Avazzia products and will present a software simulation of these implementations. The software language intended to use is C++ through Qt Creator IDE.

Test Case 1: Microcurrent biofeedback device with external sensors

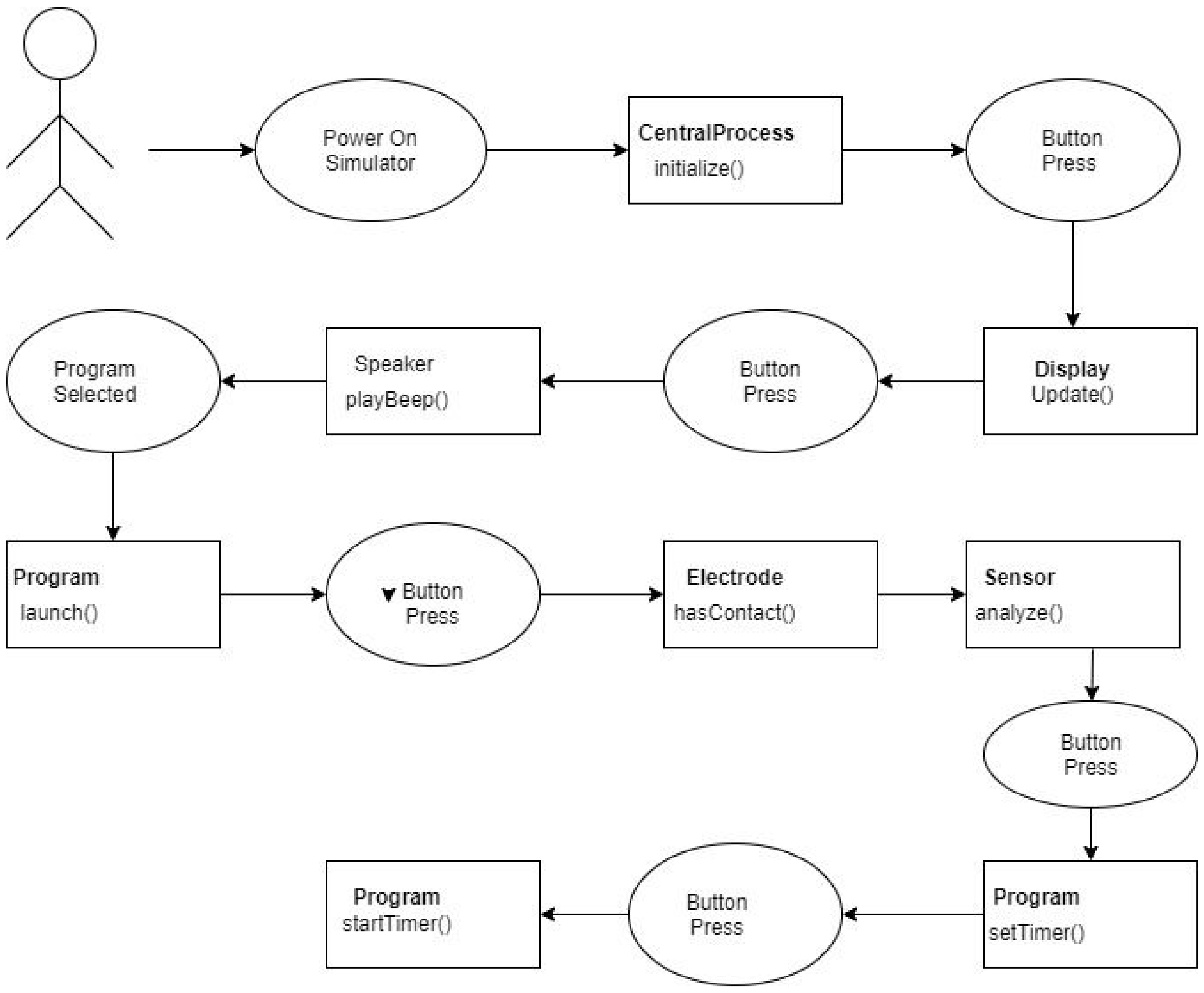


Figure 1: Device connected with external sensors

The use case that illustrates what happens when the device connected with external sensors to perform the therapy program is shown in Figure 1. The use case is continuation of the basic use case of figure 3 which shows how to launch the therapy program without external sensors added to the device.

Process flow of the use case:

At this point the CentralProcess is already running the initialize function and the desired program is chosen as well.

1. User presses button Electrode in the Menu

   The display will update the Menu.

2. User checks if the Electrode isOn() by running the isOn function.

   The electrode class is initiated from the menu. The frequency and power of the program are adjusted within this class.

3. The user applies the external sensor on the body part.

4. The user initializes hasContact function to make sure the external sensors are properly put on the body surface.

5. The user initializes the analyze function to see if the sensor is on the user's skin.

6. The user presses a button to set time in order to launch the therapy program.

7. The user starts time and the countdown screen appears to show the duration of the therapy program.

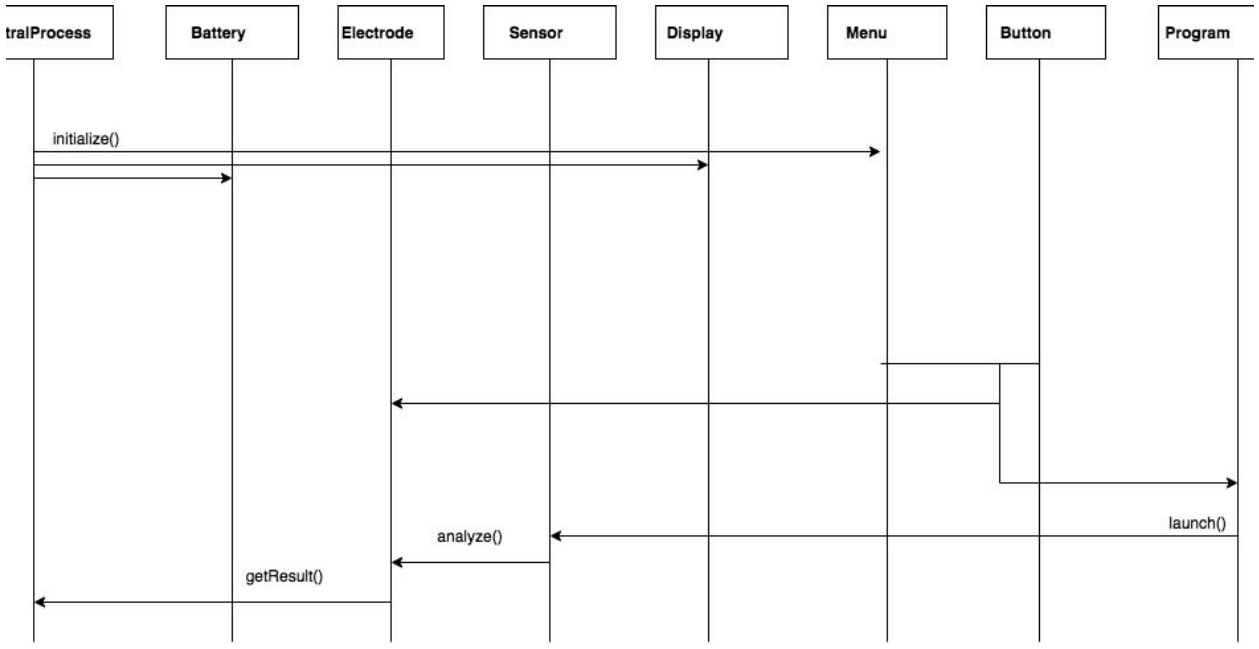


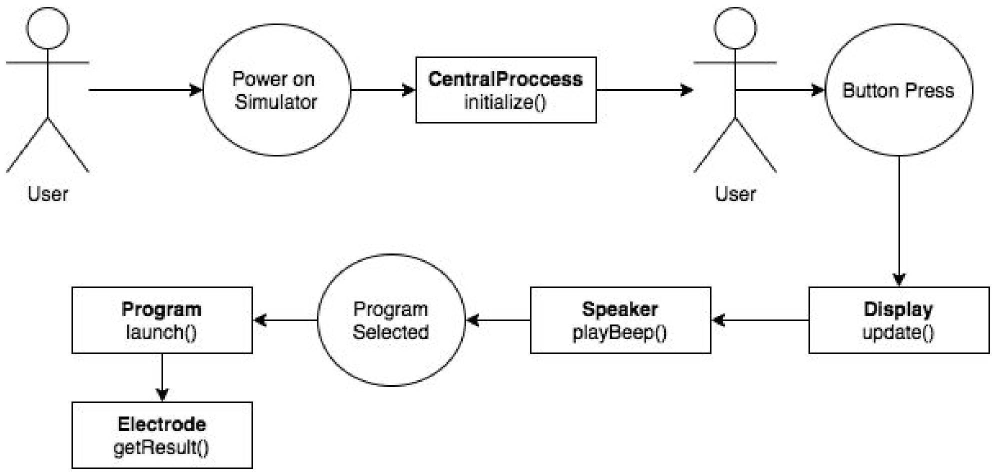
Figure 2: Sequence diagram with sensors connected

Test Case 2: Device without Sensors connected.

Use Case	**Using the simulator without external sensors**
Description	This use case describes the process a user will go through to execute a therapy session without external sensors attached
Actors	User
Triggering Event	User powers on device
Pre-Condition	The simulator is running, but powered off
Main Sequence	● User powers on device ● Presses button to indicate program selection ● Selects desired program ● Configures settings for program

	• Begins program
Post-Condition	N/A
Resulting Event	The user gets the results from the given session
Alternative Scenarios	Alternative 1: The user selects the simulator to not simulate the device on skin, which will pause the session
Comments	N/A
Traceability	?

Use Case.



The basic use case for when a user will use the device to launch a therapy program without external sensors attached to the device.

Figure 2.0

User powers on the simulator.

The CentralProccess class runs the initialize function upon power up and the Menu is updated.

User Presses a Button to move the Menu selection.

The Display is updated to highlight the new selection.

A Program is selected and a Button is pressed to select it

The Program is executed using the launch process and checks if external sensors are attached, when not detected will get the result based on skin contact therapy. Results are returned from the Electrode.

User has ended the simulator use.

This use case demonstrates the flow of an end-to-end case of use when an external electrode is not connected. It covers nearly all major processes of the simulator, apart from changing settings. The expected flow remains constant for all situations when the user is using the simulator without external sensors, and the only variable will be the returned results from the program execution.

The CentralProccess initialize function will instruct the Battery, Electrode and Menu classes to construct given default values. This ensures the device is ready for operation by the user. Once complete, the simulator will wait for input from a user before proceeding with the next step of the flow. When a button is pressed by the user, the display will update the menu with a new selection given the directionality of the button they pressed. Once the user has selected the desired program, they will press a button to select it, once again providing audio feedback. The simulator will then use the selected Program class's launch function to begin execution of a given program. Once complete the results of the session will be returned.

Sequence Diagram.

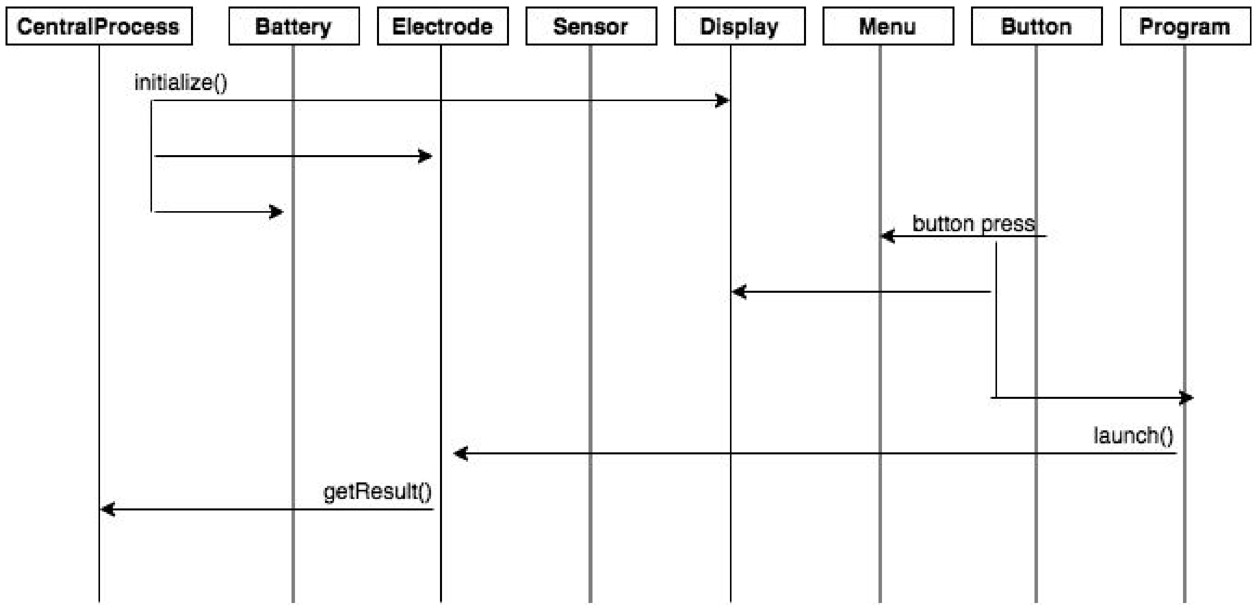


Figure 2.1

Test Case 3: User Changes Settings.

Use Case	Device user changes a setting
Actor	Patient/Doctor
Use Case Overview	The patient/doctor navigates to the “Settings” menu to alter some characteristic of the device
Trigger	The patient/doctor desires to alter a setting on the device
Precondition	The device is turned on
Assumption	The display is currently showing the main menu

Common Flow

Description	Device user navigates to the "Settings Menu"
1	User presses the "Down Arrow" button a certain number of times
2	Each time the button is pressed, the display updates the currently selected option
3	Once the current selection of the display is the "Settings" option, the user presses the "OK" button
4	The display updates to show the "Settings Menu"

Case 3.1: User changes a sound setting

Basic Case

Description	Device user changes the state of the general sound setting
1	User presses the "OK" button to select the "Sound" option
2	The display updates to show the "Sound Menu"
3	User presses the "OK" button to select the "General" option
4	The display is updated to show the level adjustment screen
5	The user presses the left and/or right buttons
6	The display updates the number on screen and the progress bar according to the "General" sound value
7	The user finds an acceptable sound level and presses "OK"
8	The display updates to show the "Sound Setting Menu"

Alternate flow 1A: Button Sound

Description	Device user changes the state of the button sound setting
3A	User presses the "OK" button to select the "Button" option

Alternate flow 1B: Contact Sound

Description	Device user changes the state of the contact sound setting
3A	User presses the "OK" button to select the "Contact" option

Alternate flow 9A: Back button

Description	Device user presses "Back" instead of "OK"
9A	User presses the "Back" button

Alternate flow X: User changes the brightness instead of sound

Description	Device user changes the state of the brightness setting
1X	User navigates to the "Brightness" option

6X	For every press, the display's "Brightness" variable is adjusted up (right) or down (left).

Case 3.2: User toggles the economy setting

Basic Case

Description	Device user turns economy mode on
1	User navigates to the "Economy" option and presses "OK"
2	The display is updated to show the "Economy Menu"
3	The user presses "OK" to select "On"
4	The Central System's member "Economy mode" is set to true

Alternate flow A: Toggle off

Description	Device user turns economy mode off
3A	The user navigates to the "Off" option and presses "OK" to select it
4A	The Central System's member "Economy mode" is set to false

Case 3.3: User alters child mode

Basic Case

Description	Device user picks an age range to enable child mode for
1	User navigates to the "Child Mode" option and presses "OK" button
2	The display is updated to show the "Child Mode Menu"
3	The user navigates to the desired age range and presses "OK"
4	The central system's child mode member is updated to the corresponding age range

Alternate flow A: Toggle off

Description	Device user turns child mode off
3A	The user navigates to the "Off" option and presses "OK" to select it
4A	The Central System's member "Child mode" value is set to null

Case 3.4: User changes the time

Basic Case

Description	Device user changes the current time on the device
1	User navigates to the “Clock” option and presses the “OK” button
2	The display is updated to show the current time
3	The user presses the “OK” button
4	The display updates to show the hour value is now flashing
5	The user adjusts the hour using the right (increase the value) and left (decrease the value) buttons
6	The display updates the number on screen
7	The user finds an acceptable hour value and presses “OK”
8	The clock’s hour value is updated
8	The display updates to show the minute value is now flashing
9	The user adjusts the minute value using the right (increase the value) and left (decrease the value) buttons
10	The display updates the number on screen
11	The user finds an acceptable minute value and presses “OK”
12	The clock’s minute value is updated

Case 3.5: User sets the alarm clock

Basic Case

Description	Device user changes the time the alarm is set for
1	User navigates to the “Alarm clock” option and presses the “OK” button
2	The display is updated to show the “Alarm Clock Menu”
3	The user navigates to the “Time” option and presses “OK”
4	The user changes the time the alarm is set for as in case 3.5 steps 3-12

Alternate flow A: Toggle off

Description	Device user turns the alarm clock off
3A	The user navigates to the “Off” option and presses “OK” to select it
4A	The Clock’s member “frequency” is set to null

Alternate flow B: Change frequency

Description	Device user changes the frequency off the alarm clock
3B	The user navigates to the “Frequency” option and presses the “OK” button
4B	The display updates to show the “Alarm frequency menu”
5B	The user navigates to the desired frequency (ie. daily, every other day, ,weekly, monthly) and presses “OK”
6B	The Clock’s member “frequency” is set to the choice of the user

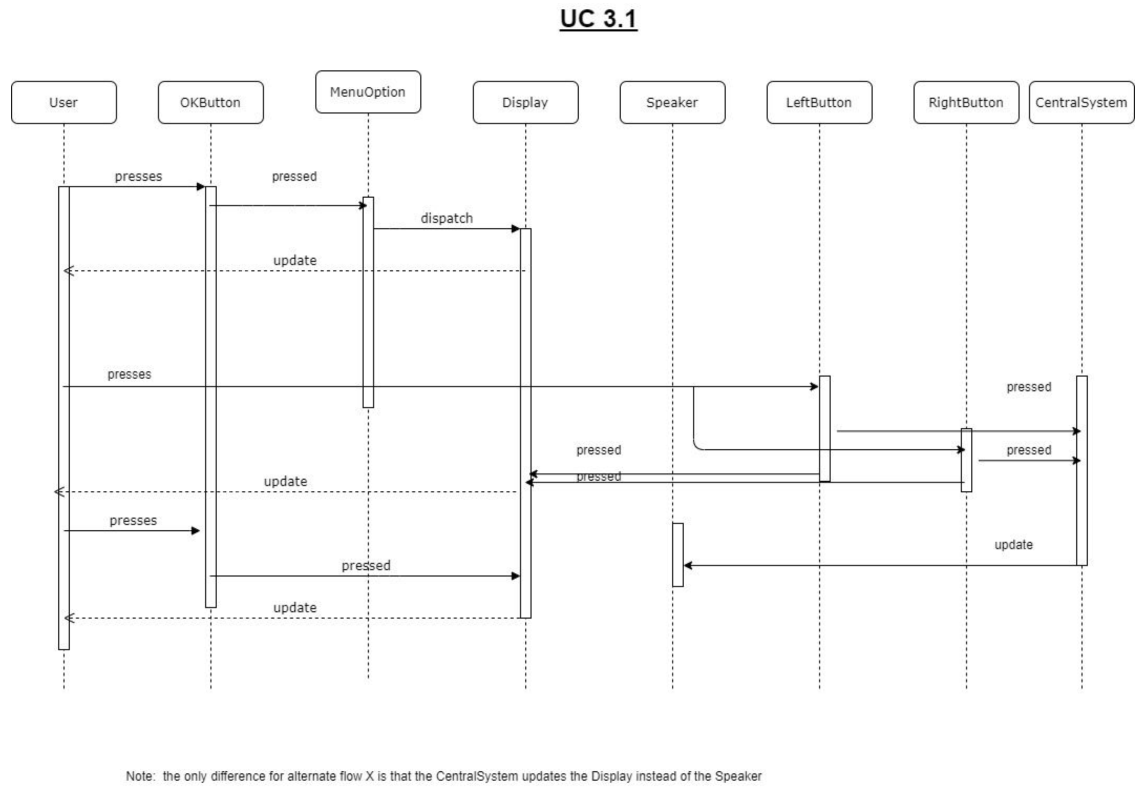


Figure 3.1

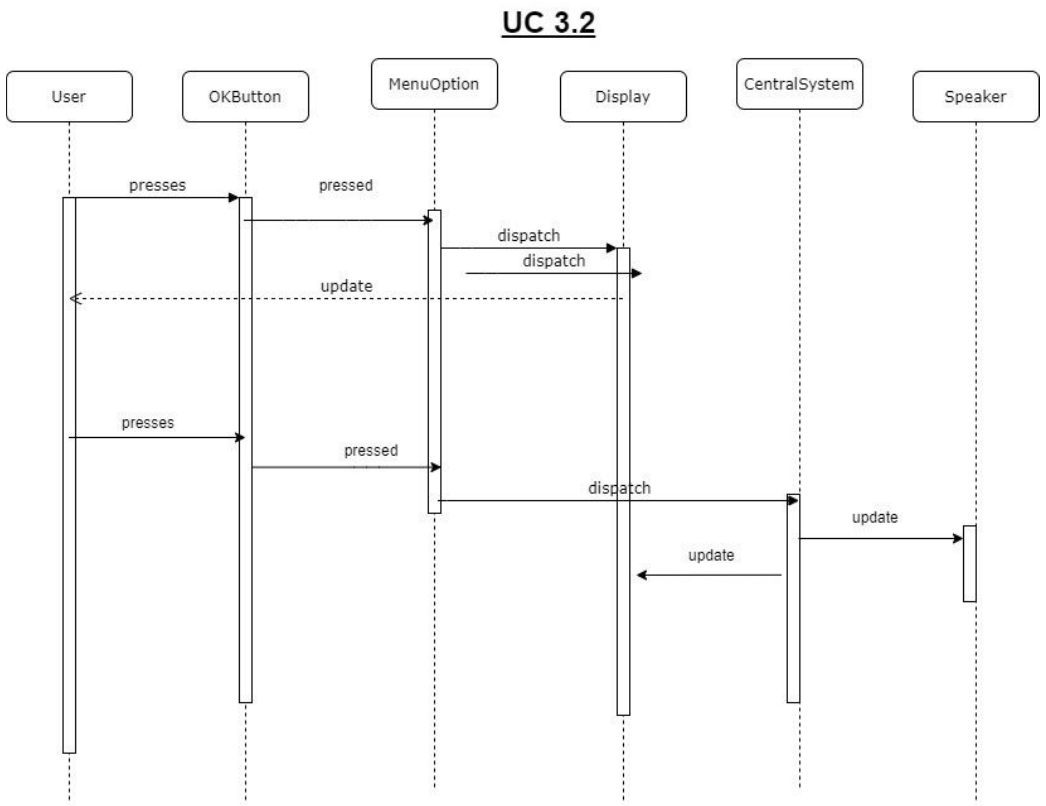
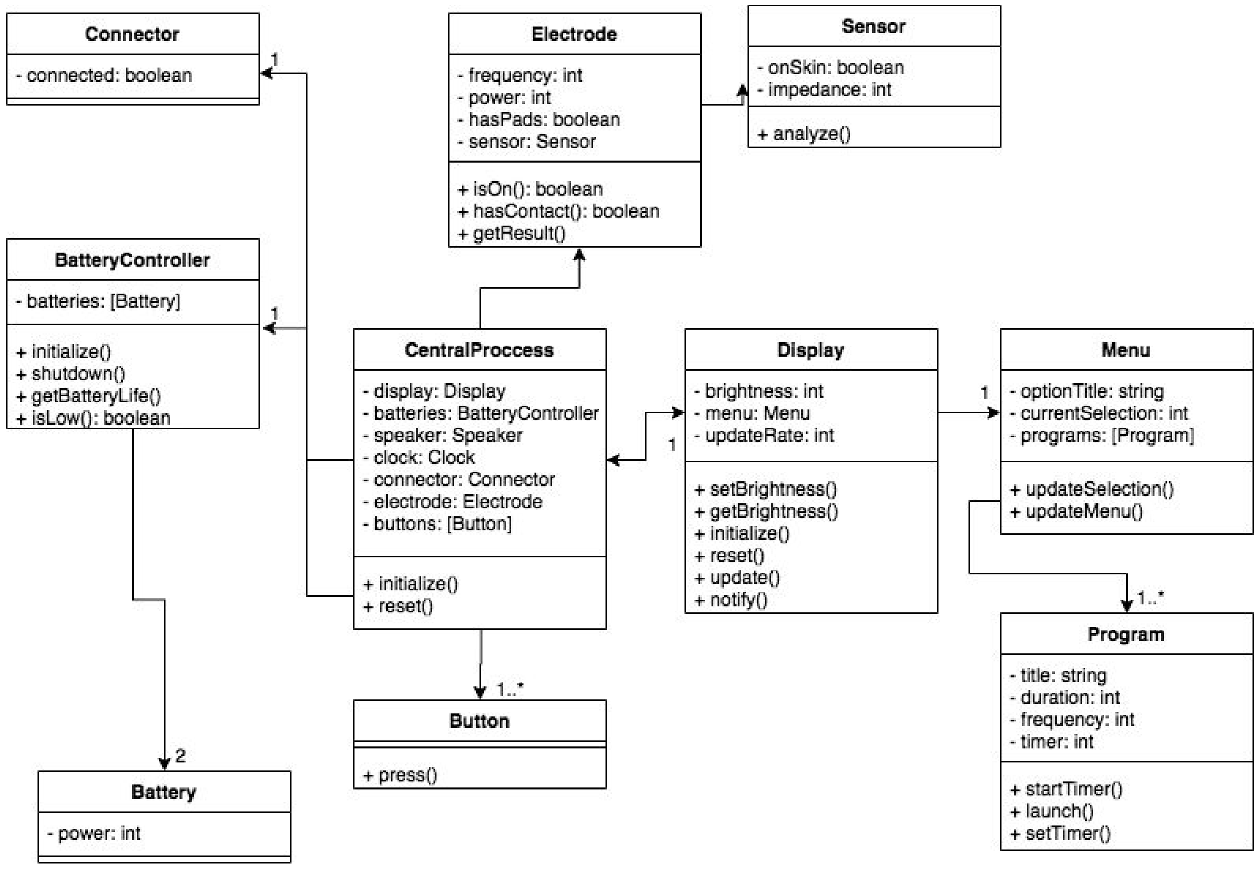

UC 3.2
User
OKButton
MenuOption
Display
CentralSystem
Speaker
presses
pressed
dispatch
dispatch
update
presses
pressed
dispatch
update
update

Figure 3.2

Overall UML Class Diagram:

Listed below is the overall UML class diagram consisting of all of the listed classes and the interactions between each other.


Connector
- connected: boolean
1
Electrode
- frequency: int
- power: int
- hasPads: boolean
- sensor: Sensor
+ isOn(): boolean
+ hasContact(): boolean
+ getResult()
Sensor
- onSkin: boolean
- impedance: int
+ analyze()
BatteryController
- batteries: [Battery]
+ initialize()
+ shutdown()
+ getBatteryLife()
+ isLow(): boolean
1
CentralProccess
- display: Display
- batteries: BatteryController
- speaker: Speaker
- clock: Clock
- connector: Connector
- electrode: Electrode
- buttons: [Button]
+ initialize()
+ reset()
Display
- brightness: int
- menu: Menu
- updateRate: int
+ setBrightness()
+ getBrightness()
+ initialize()
+ reset()
+ update()
+ notify()
1
Menu
- optionTitle: string
- currentSelection: int
- programs: [Program]
+ updateSelection()
+ updateMenu()
1
1..*
Program
- title: string
- duration: int
- frequency: int
- timer: int
+ startTimer()
+ launch()
+ setTimer()
1..*
Button
+ press()
2
Battery
- power: int