

# 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

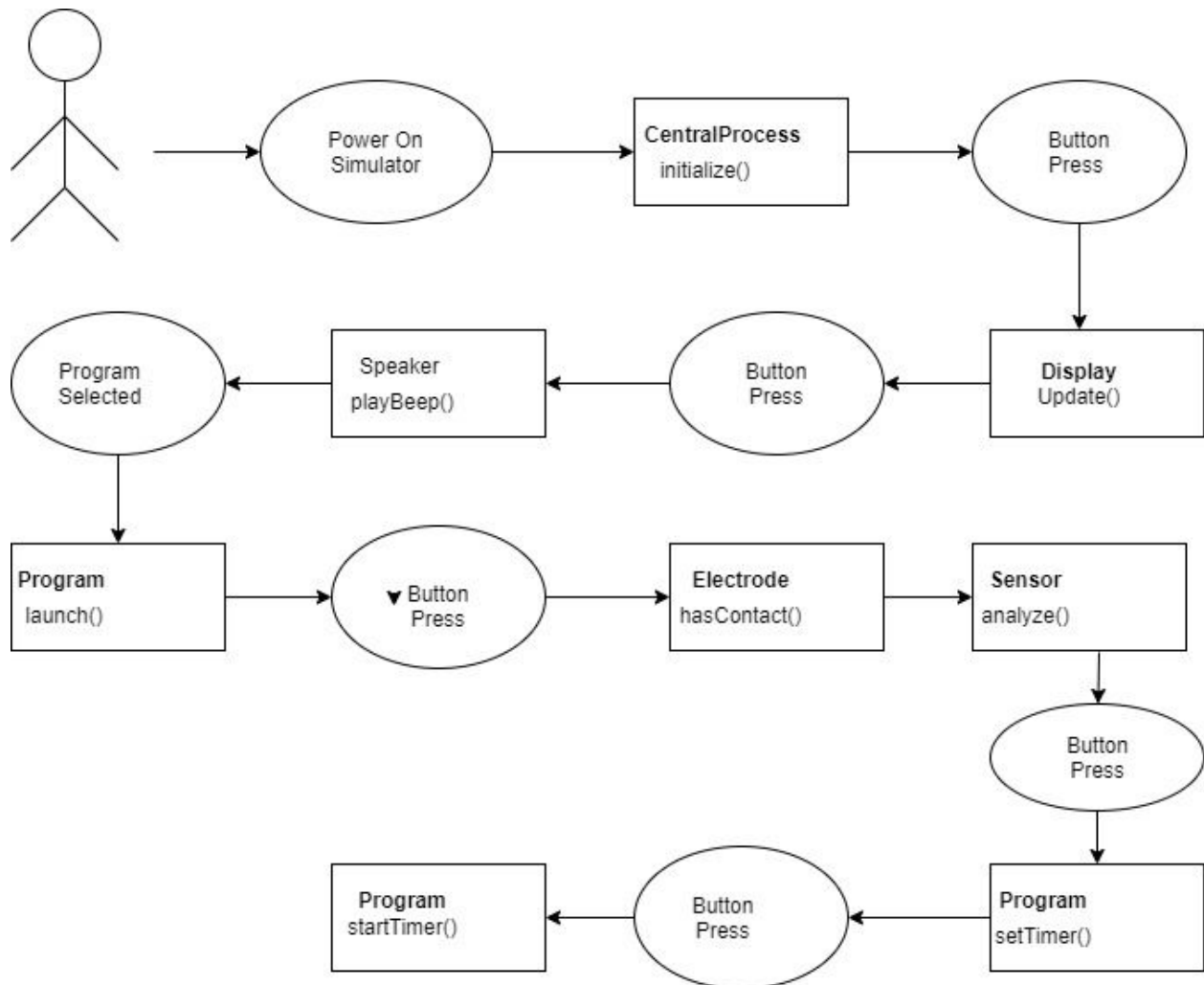


Figure 1: Device connected with external sensors

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.

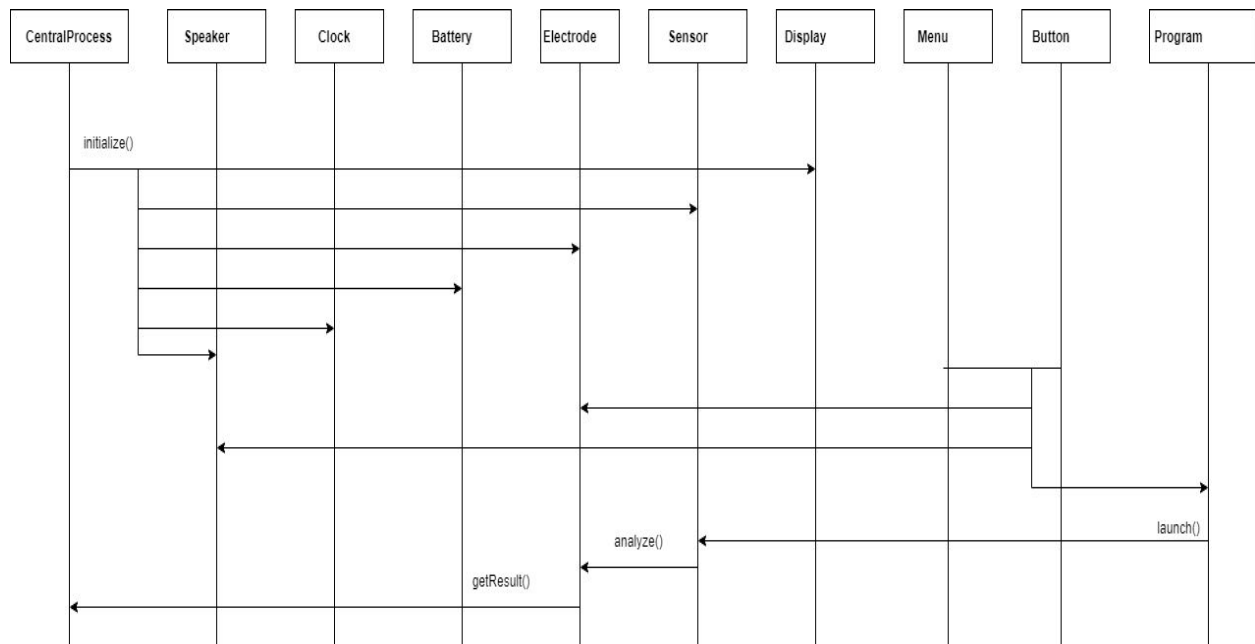


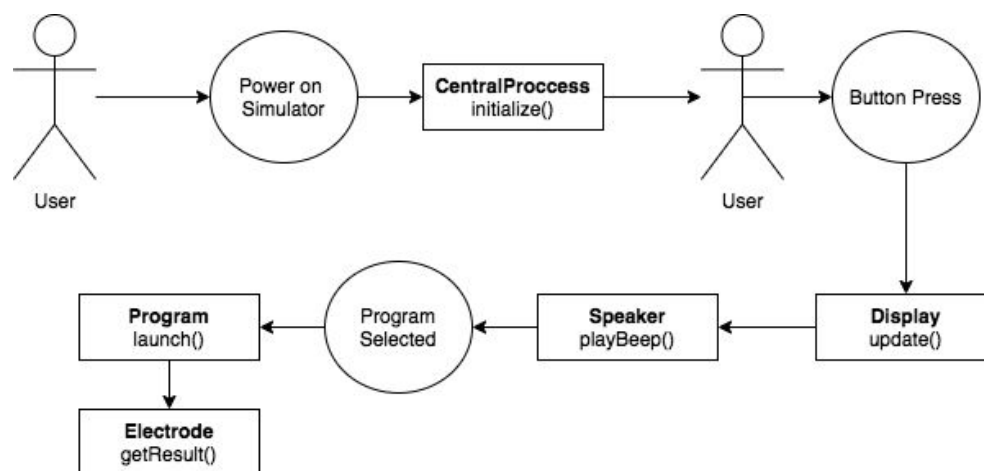
Figure 2: Sequence diagram with sensors connected

## Test Case 2: Device without Sensors connected.

<b>Use Case</b>	<b>Using the simulator without external sensors</b>
<b>Description</b>	This use case describes the process a user will go through to execute a therapy session without external sensors attached
<b>Actors</b>	User
<b>Triggering Event</b>	User powers on device
<b>Pre-Condition</b>	The simulator is running, but powered off
<b>Main Sequence</b>	<ul style="list-style-type: none"><li>• User powers on device</li><li>• Presses button to indicate program selection</li><li>• Selects desired program</li><li>• Configures settings for program</li></ul>

	<ul style="list-style-type: none"><li>• Begins program</li></ul>
<b>Post-Condition</b>	N/A
<b>Resulting Event</b>	The user gets the results from the given session
<b>Alternative Scenarios</b>	Alternative 1: The user selects the simulator to not simulate the device on skin, which will pause the session
<b>Comments</b>	N/A
<b>Traceability</b>	?

*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 CentralProcess 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 Beep is played through the speakers to indicate that a selection has been made.**

*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 CentralProcess initialize function will instruct the Speaker, Clock, 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. The Speaker class will play a sound, notifying the user that a new selection has been made as a means of haptics and feedback to indicate a successful action. 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.*

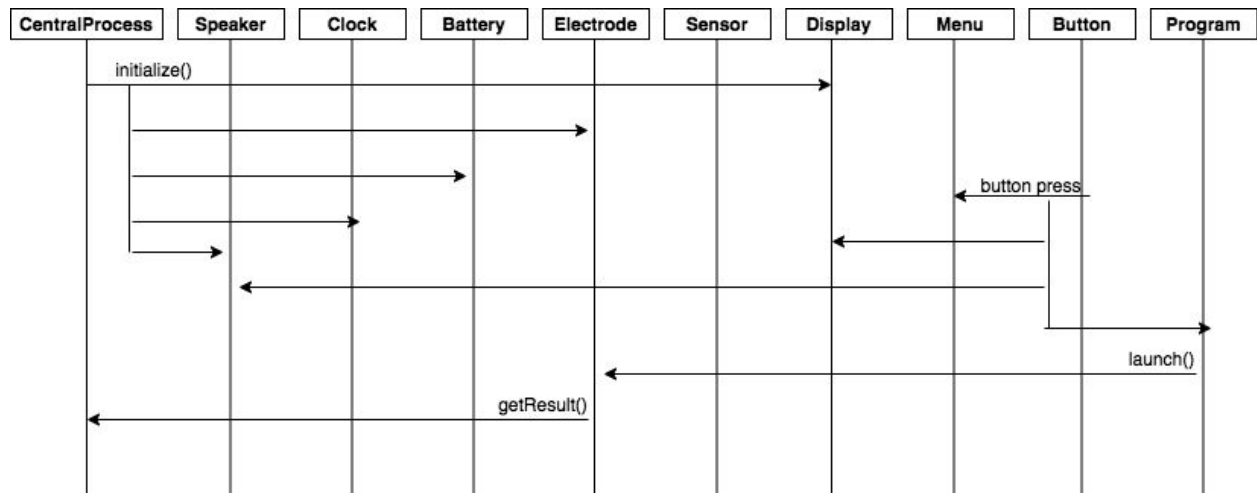


Figure 2.1

**Test Case 3: User Changes Settings.**

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

*Common Flow*

<b>Description</b>	Device user navigates to the “Settings Menu”
<b>1</b>	User presses the “Down Arrow” button a certain number of times
<b>2</b>	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	For every press, the speaker’s “General” volume is adjusted up (right) or down (left).
7	The speaker beeps for each button press.
8	The display updates the number on screen and the progress bar according to the “General” sound value
9	The user finds an acceptable sound level and presses “OK”
10	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*

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

***Case 3.5: User sets the alarm clock***

*Basic Case*

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

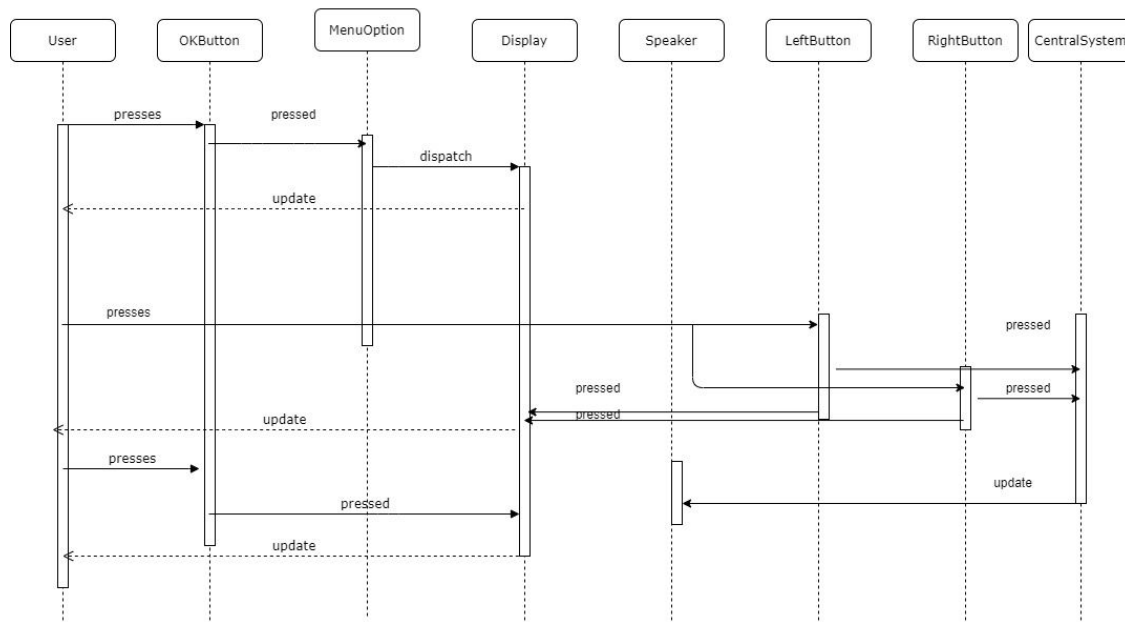
*Alternate flow A: Toggle off*

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

**Alternate flow B: Change frequency**

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

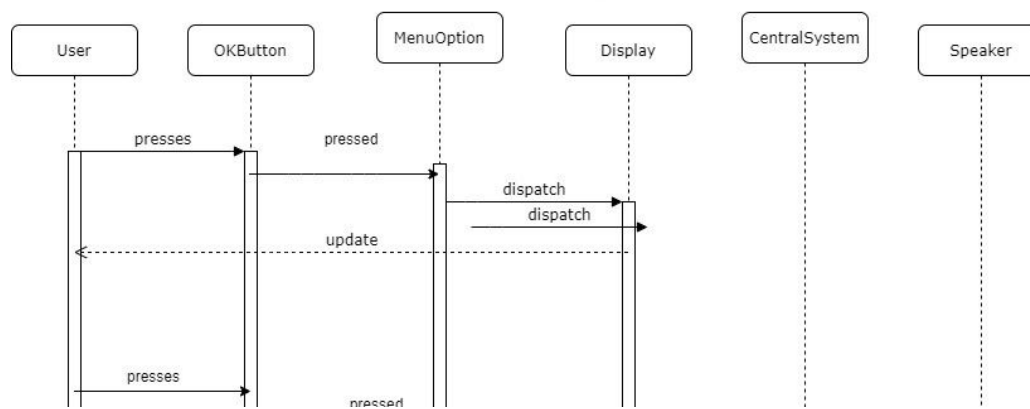
**UC 3.1**



Note: the only difference for alternate flow X is that the CentralSystem updates the Display instead of the Speaker

*Figure 3.1*

**UC 3.2**



*Figure 3.2*