COMP 3004 Final Project

Team 11 Members:

Billal Ghadie - 100888260

Oluwademilade Edward Akapo - 101095403

Michael Grewal - 100739181

Huzaifa Patel - 101117549

Developing and Testing a CES Device Simulator

The goal of this project was to develop and test a simulator for the embedded software used in Cranial Electrical Stimulation (CES) devices, similar to the Alpha-Stim product. A CES device is a non-invasive neuro-stimulation medical device that delivers microcurrents via electrodes connected to the earlobes to stimulate the brain for the purposes of therapeutic intervention. This non-invasive intervention has been shown to treat conditions such as insomnia, anxiety, and depression. The implementation was done in C++ using the QT framework. What follows are the use cases that were designed from the given specification.

Use Cases

UC1: Starting the Therapy Session

Primary Actor: Patient

Goal in Context: The patient turns on and configures the AlphaStim. The therapy session itself is excluded.

Scope: The setup of the AlphaStim.

Level: User goal

Stakeholders and Interests:

Patient - Wants to start the therapy session and have their symptoms alleviated.

Manufacturer - Wants the AlphaStim to turn on and function as designed.

Minimal Guarantees: The AlphaStim will provide therapy for the described symptoms provided the batteries are sufficiently charged.

Success Guarantees: The AlphaStim is turned on and configured as desired.

Trigger: A patient wishes to begin a therapy session.

Main Success Scenario:

1. **Patient**: presses the AlphaStim's on button.
2. **Patient**: connects an electrode electro to each ear.
3. **Patient**: Selects one of the three frequency options.
4. **Patient**: Selects the desired waveform.
5. **Patient**: Selects the duration of the therapy session.
6. **Patient**: Selects the desired current intensity using the current control.

7. **Patient**: Presses the start button.
8. **AlphaStim**: Updates the timer display and begins the therapy.

Extensions:

1a. At any point the battery level could diminish to a critical level, see *UC3: Receiving Low Battery Alerts*.

1b. At any point the AlphaStim could fault and the current exceed 700 μa, see *UC6: Auto disable when current exceeds 700* μa.

2a. At any point the electrodes could be disconnected from the patient's ear(s), see *UC7: Check contact with skin*.

UC2: Adjusting the Current

Primary Actor: Patient

Goal in Context: The patient changes the microampere of the current intensity to a desired level. Starting the therapy and whether the therapy is being recorded is excluded.

Scope: Adjusting the current intensity of the device.

Level: User goal.

Stakeholders and Interests:

Patient - Wants to adjust the current intensity during the therapy session.

Manufacturer - Wants the AlphaStim to adjust the current intensity appropriately.

Minimal Guarantees: The AlphaStim will provide therapy for the described symptoms provided the batteries are sufficiently charged.

Success Guarantees: The current intensity is adjusted.

Trigger: The patient wants to change the current intensity.

Main Success Scenario:

1. **Patient**: *Starts the Therapy Session (UC1).*
2. **Patient**: uses the up/down button to change the current being administered.
3. **AlphaStim**: adjusts the current intensity being administered.
4. **AlphaStim**: updates the current intensity display.

Extensions:

1a. At any point the battery level could diminish to a critical level, see *UC3: Receiving Low Battery Alerts*.

1b. At any point the electrodes could be disconnected from the patient's ear(s), see *UC7: Check contact with skin*.

1c. At any point the AlphaStim could fault and the current exceed 700μa, see *UC6: Auto disable when current exceeds 700*μa.

UC3: Receiving Low Battery Alerts

Primary Actor: Patient

Goal in Context: The patient is given a low battery alert when at 5%, and another at 2%, followed by a shutdown.

Scope: The AlphaStim device.

Level: User goal

Stakeholders and Interests:

Patient - wants to know when battery level is low and soon to be shut down.

Manufacturer - wants the user to know if they have enough charge for treatment, or if they should replace the battery.

Minimal Guarantees: Same as Success Guarantees.

Success Guarantees: The AlphaStim emits low battery alerts at 5% and 2%, and the device shuts down at 2%.

Trigger: The AlphaStim is powered on and the battery level reaches 5% (first warning) and 2% (second warning then shut down).

Main Success Scenario:

1. **Patient:** is using the AlphaStim.
2. **AlphaStim:** battery is consumed as the device is in use.
3. **AlphaStim:** the battery level reaches 5%.
4. **AlphaStim*:*** the first low battery warning is given to the patient.
5. **Patient**: continues using the device.
6. **AlphaStim:** the battery level reaches 2%.
7. **AlphaStim**: the second low battery warning is given to the patient.

8. **AlphaStim:** the AlphaStim device shuts down.

Extensions:

3a. The battery level is above 5%:

3a1. No need to warn the patient because there is sufficient charge remaining.

5a. The patient does not continue using the device:

5a1. The battery will resume from the current charge level the next time the patient begins therapy.

5a2. The patient may decide to replace the battery before their next therapy session.

8a. The AlphaStim fails to shut down:

8a1. The device will shut down regardless if battery level reaches 0%. However, the patient runs the risk of losing their current therapy session record.

UC4: Recording a Therapy Session

Primary Actor: Patient

Goal in Context: The patient commands the AlphaStim to store session data.

Scope: The AlphaStim device

Level: User goal

Stakeholders and Interests:

Patient - wants to be able to review therapy session history.

Manufacturer - wants to build the AlphaStim with functionality to view therapy session history on the device itself.

Minimal Guarantees: Therapy session record is stored in the AlphaStim's memory and can be viewed later.

Success Guarantees: If the patient chooses to store therapy session records, then the AlphaStim will store the record in memory and allow the patient to view all records whenever they want.

Trigger: Before the patient begins a therapy session, the AlphaStim prompts the patient if they would like this session to be saved.

Main Success Scenario:

1. **Patient:** *Starts the Therapy Session (UC1)*.
2. **AlphaStim:** Prompts the patient if they would like to save this session's data.
3. **Patient:** Replies "Yes" to the AlphaStim's prompt.
4. **AlphaStim**: Begins and finishes therapy session.
5. **AlphaStim**: Saves the therapy session record to memory.

Record: {timestamp (yyyy-mm-dd, time), waveform, frequency, max intensity, duration}

Extensions:

1a. Patient is not ready to begin:

1a1. Therapy does not start and record will not be saved.

2a. AlphaStim fails to prompt the patient:

2a1. For security, record will not be saved unless the patient explicitly agrees.

3a. The patient replies “No”:

3a1. The record will not be saved.

4a. The therapy session is interrupted:

4a1. The incomplete therapy session data is stored.

5a. Insufficient memory available:

5a1. The therapy session is not recorded.

UC5: Auto Sleep When Not In Use

Primary Actor: AlphaStim Device

Goal in Context: The device automatically turns off when it has not been used for 30mins.

Scope: AlphaStim

Level: Device operation

Stakeholders and Interests:

Patient: wants the battery of their device to last long

Manufacturers: wants the device battery to be optimally managed when not in use

Minimal Guarantees: Same as Success Guarantees.

Success Guarantees: Device correctly measures time since last use and switches off all components.

Trigger: Patient has stopped using device

Main Success Scenario:

1. **AlphaStim**: Measures time elapsed since inactivity.
2. **AlphaStim**: When time reaches 30 mins the device is set to sleep state.
3. **AlphaStim**: While the device is in sleep state all components are turned off.

UC6: Auto Disable When Current Exceeds 700µa

Primary Actor: AlphaStim device

Goal in Context: Device automatically permanently disables if a fault causes the current to exceed 700µa. The source of the fault is not included.

Scope: AlphaStim device.

Level: Device operation

Stakeholders and Interests:

Patient: wants their device to be safe to use

Manufacturers: wants the device to operate within the specific current constraints

Minimal Guarantees: Same as Success Guarantees.

Success Guarantees: AlphaStim accurately senses the erroneous increase in current and shuts off permanently.

Trigger: AlphaStim current sensor receives a current exceeding 700µa

Main Success Scenario:

1. **AlphaStim**: current sensing resistor receives a current exceeding 700µa.
2. **AlphaStim**: processes signals then sets the device to a permanent off state.

UC7: Check Contact With Skin

Primary Actor: AlphaStim device

Goal in Context: flash circuit symbol and pause/resume timer when skin contact with patient changes

Scope: AlphaStim device

Level: Device operation

Stakeholders and Interests:

Patient: wants to be notified if skin contact is not detected.

Manufacturer: only wants data to be recorded when probes are in contact with skin

Minimal Guarantees: Same as Success Guarantees.

Success Guarantees: Device accurately senses changes in skin contact to flash the symbol and pause and resume the timer.

Trigger: Patient changes connection of electrodes with skin.

Main Success Scenario:

1. **AlphaStim**: probes sense whether there is skin contact with a patient and send signals to the device.
2. **AlphaStim**: if there is contact with the skin resume timer and remove symbol.

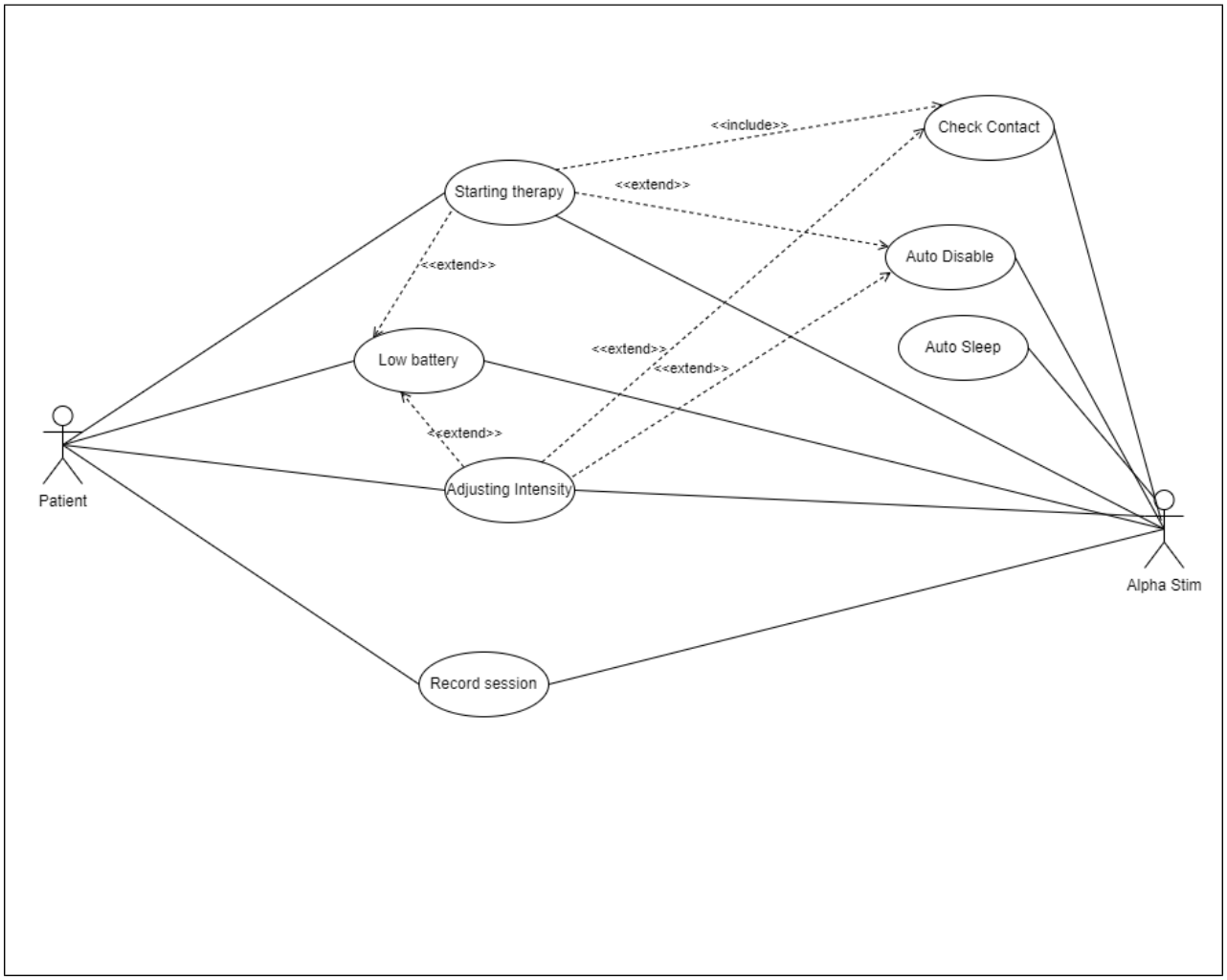
Extensions:

2a. If there is no contact with skin.

2a1. The timer is paused and the test circuit symbol is flashed.

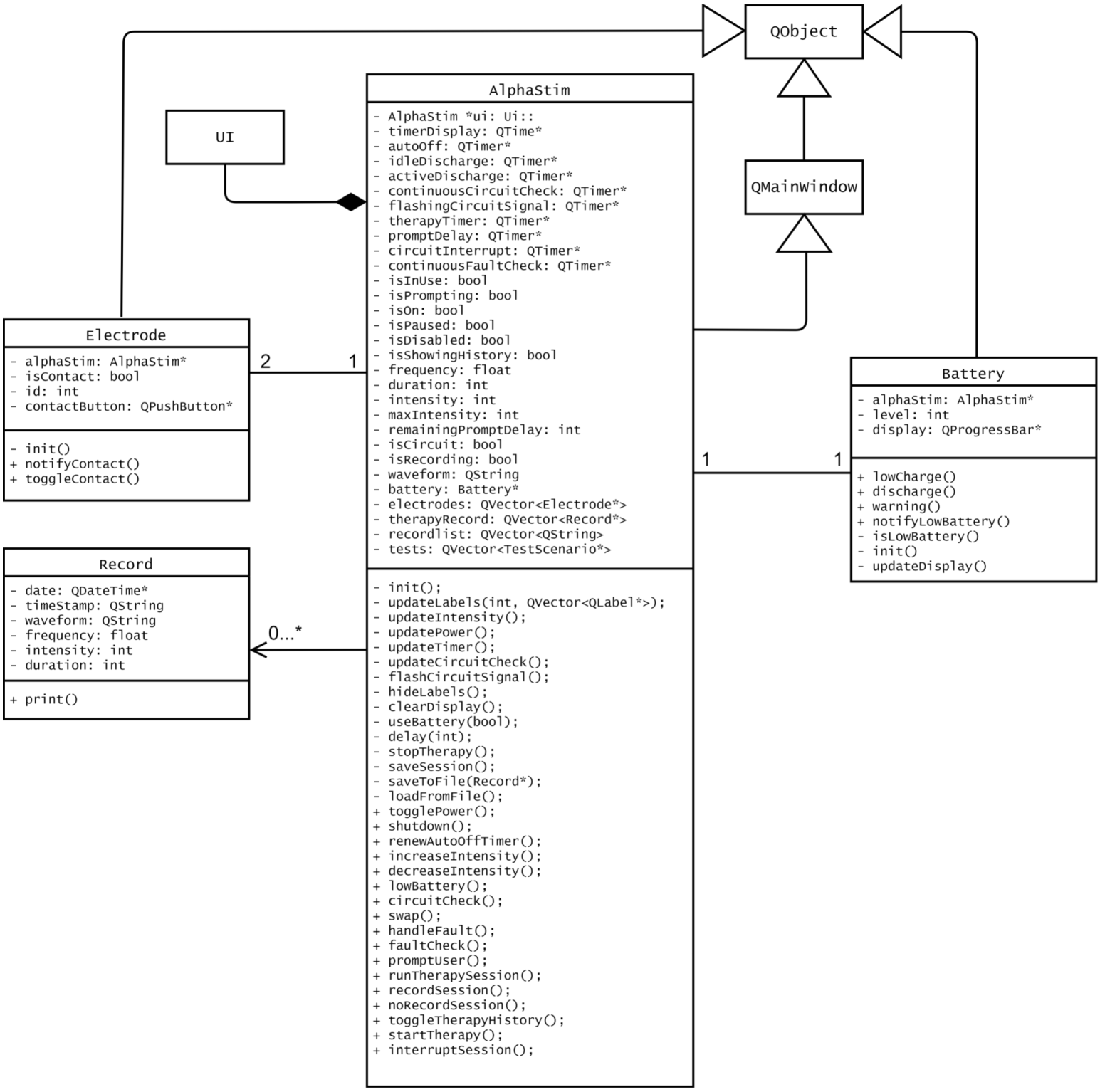
Use Case Diagram

Use Case Diagram For Alpha Stim Device



UML Class Diagram

A structural model representing the AlphaStim system.



Explanation of UML Class Diagram

The UML class diagram illustrates the observer pattern with the AlphaStim acting as the subject, and the battery and electrodes acting as observers. The C++ QT framework is used to create the CES simulation by modeling the AlphaStim and its components, which include the “Battery”, “Record”, and “Electrode” classes.

AlphaStim Class

Out of all the classes contained in the source code, the AlphaStim class is the most significant. The AlphaStim class orchestrates the main program flow, has the ability to control or manipulate almost every configuration of the device (i.e. intensity, waveform, duration, etc...), and communicates with every other entity class in the source code. For example, the AlphaStim class communicates with one battery to get information on its power level. Second, the AlphaStim class interacts with zero or more records so that it can show them on the device's display. Last, the AlphaStim class communicates with two electrodes to determine if the electrodes are in contact with the patient's skin. The AlphaStim class is also a child of QMainWindow and QObject, which gives it the ability to both interact with the user interface (UI) provided by the C++ QT framework, as well as use all of the functions provided by the QMainWindow and QObject classes. More formally, the AlphaStim class has a “has a” relationship with the Battery, Record, and Electrode classes. Additionally, the AlphaStum class has a “is a” relationship with the QMainWindow and QObject classes.

Battery Class

The Battery class is an entity class that is responsible for monitoring the battery’s power level, and informing the AlphaStim of a low battery state. The Battery class will emit a warning notification to the AlphaStim when its power level is low, and regularly update the user interface

(UI) with its charge level. The Battery class is able to interact with the UI because it has a “has a” relationship with the AlphaStim class. This design decision is explained in more detail under the *Design Decisions* section.

Record Class

The Record class is an entity class that is responsible for keeping track of the records that the patient decides to save. The record class is a unique class because it is the only class that does not have a “has a” or “is a” relationship with any other class.

Electrode Class

The Electrode class is an entity class that is the point of contact between the AlphaStim and the patient. It provides reliable information to the AlphaStim class about whether it is making contact with the patient’s skin or not. The Electrode class is able to receive signals from the UI because it has a “has a” relationship with the AlphaStim class. Due to this, it has a QPushButton member which toggles contact with skin. The Electrode class is aware when this button is pressed, and is able to interact with the AlphaStim class by consequently emitting update signals when contact is made or broken.

Composition Link

The UML Class Diagram uses the composition link because in addition to the “has a” relationship between the AlphaStim class and the UI class, there's a strong lifecycle dependency between the two. More specifically, when the AlphaStim class is destructed, then the UI class will also be destructed.

QMainWindow & QObject

The QMainWindow is a child of QObject. Both of these classes are part of the QT framework, and they are both essential for developing an observer pattern, as well as creating the User Interface (UI).

Design Decisions

Observer Pattern

Our AlphaStim application implements the observer pattern. Components such as the battery and electrodes act as observers to the AlphaStim, which serves as the subject, as well as the UI. This design effectively decouples the central unit from its peripherals. For example, when an electrode detects contact with the patient's skin it updates the AlphaStim, informing it of a connection made. Multiple electrodes need not worry about any other electrodes because the AlphaStim will handle subsequent events depending on the state of all its components. In other words, when events are triggered in the UI, a notification signal is broadcasted, any class which is listening will invoke a specific method, followed by an update of state change and the main program flow continues. This is precisely how we would expect the observer pattern to perform.

From a software developers perspective, we wanted to achieve a higher degree of encapsulation for the battery. Therefore, it's behaviour is handled in it's own class, contrary to how a battery behaves in real life. In actuality, a battery does not have a memory, it does not know if it is depleted or charged, and it is not capable of communicating. However, we felt it was a wise design decision to have the battery's behaviour written in it's own class and playing the role of an observer to the AlphaStim. As the AlphaStim consumes the battery, the battery will update the AlphaStim when it's charge reaches 5% and 2% (written as LOW_LEVEL and CRITICAL_LEVEL constants, respectively), then the AlphaStim handles the low charge state by showing warning messages to the display screen.

Defined Constants and Dynamic Programming

The application is programmed to be dynamic, through the use of defined constants in the source code. Many parameters can be tweaked to change how the simulation behaves and

reacts, with the exception of creating the UI element for it. For example, add more electrodes by simply creating another button and incrementing the NUM_ELECTRODES. Add more test cases by simply creating another test button and incrementing the NUM_TEST_SCENARIOS. Tune the auto off timer by adjusting the AUTO_OFF_TIMER if 30 minutes is too long or too short. Change how fast the battery drains by adjusting the BATTERY_USAGE_RATE. Alter the speed of time by changing the THERAPY_DECREMENT_RATE. Easily add or remove frequencies, durations, and waveforms by adjusting their respective constants. Change the low battery level warnings to any other percentage by adjusting the CRITICAL_LEVEL and LOW_LEVEL constants, and much more.

Writing our program like this is good software engineering, because it promotes code reliability and reusability with minimal intervention. Our decision allows the manufacturer to easily tweak many parameters to their preference. If the manufacturer wants to release revisions of the product, then it is much easier and safer to change a constant than it is to rewrite and change dozens of lines of code. Therefore, this design decision improves code quality and reusability.

Further, this design decision could save a lot of money by reducing the hours spent coding, testing, manufacturing, and experimenting under different parameters. It enables the research and development team to easily explore new conditions which helps drive future releases of the product.

Saving History and Persistent Storage

The record class is an entity class which is responsible for storing individual therapy session information. The data stored includes date, time, waveform, frequency, max intensity of the session and duration elapsed. Encapsulation and abstraction is achieved by separating the record data from the main program flow.

Upon launching the application, a save file is loaded into memory which contains any saved therapy sessions. If the save file does not exist, then a new one is created. The file is not

encrypted and is stored as plain text. We deemed that the therapy session history was not sensitive information, and does not compromise any personal data, therefore plain text would suffice.

We accomplished this goal by using the interfaces provided by the QFile and QTextStream classes. A record (if chosen to be saved) is essentially printed to the save file, and can be loaded later by reading.

Test Architecture

The TestScenario class provides an extensive and robust interface for conducting test cases. Each test can be executed by simply calling it's run() function, which triggers a specific script depending on it's scenario number (e.g. scenario *n* will run test script *n*). The tests log their progress to the console, and continuously perform state checking to determine if the result is a pass or fail. The continuous state checking verifies that the actual state of whatever is being tested matches the expected state. If there is a mismatch at any stage then the test is a fail, otherwise, it's a pass.

Speed of Time

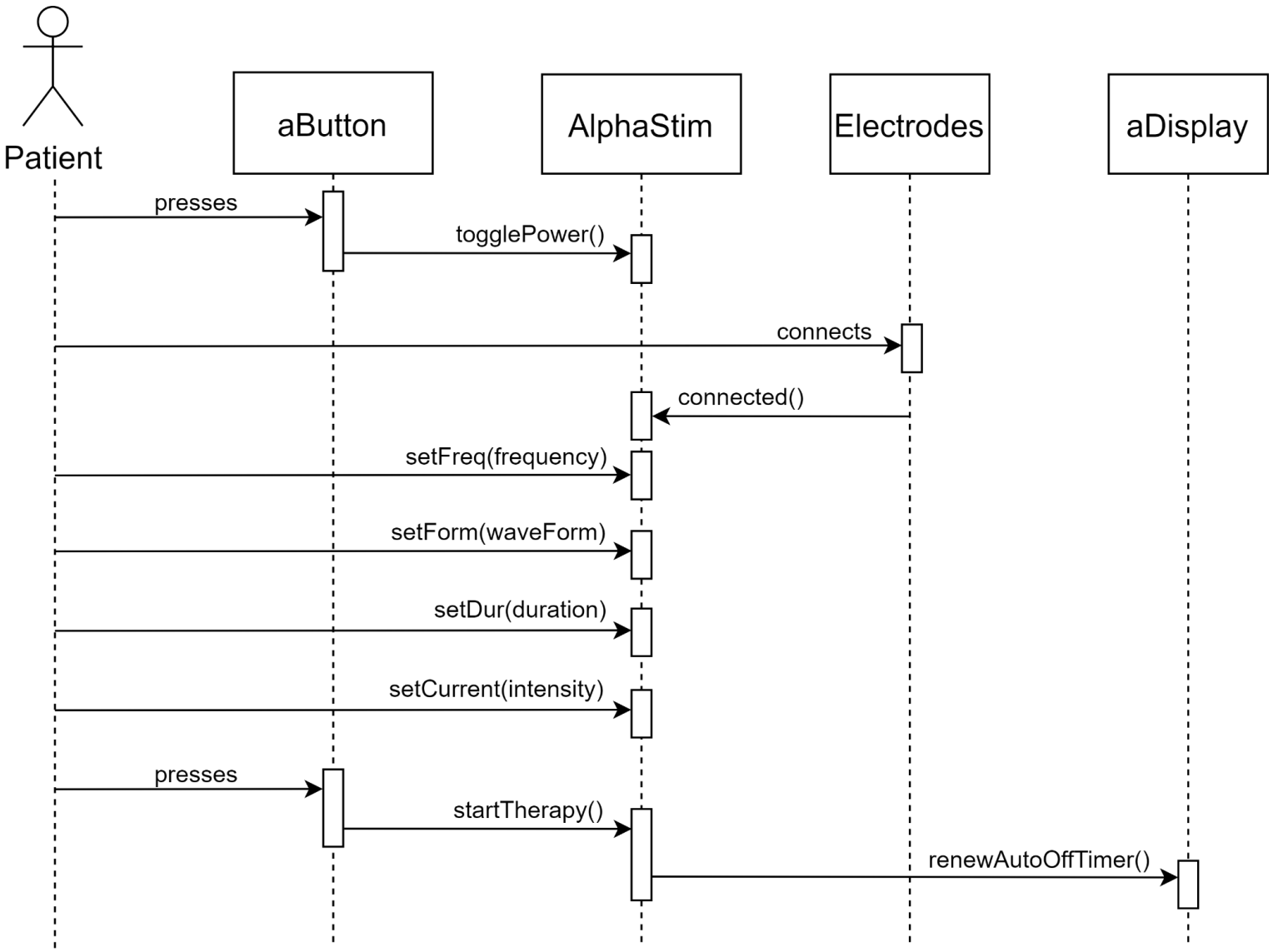
Out of the box, the speed of time is not 1:1 as it behaves in real life. We sped up the expiry rate of the therapy timer to make the simulation more user friendly and less time consuming for testers and demos. For example, 1 second in the simulation is actually 15 milliseconds, and the rate can be adjusted with the THERAPY_DECREMENT_RATE constant. The 30 minute auto off feature is actually 30 seconds in the simulation because waiting for 30 minutes is not realistic for demonstrations and testers. Nonetheless, the application is modelling this feature in the simulation, regardless of the duration. The duration can be adjusted easily by changing the AUTO_OFF_TIMER constant in alphastim.h. To model 30 minutes, then simply set the AUTO_OFF_TIMER to 1,800,000 milliseconds.

UML Sequence Diagrams

UML Sequence Diagram for UC1

Starting from the top left of the UML Sequence Diagram:

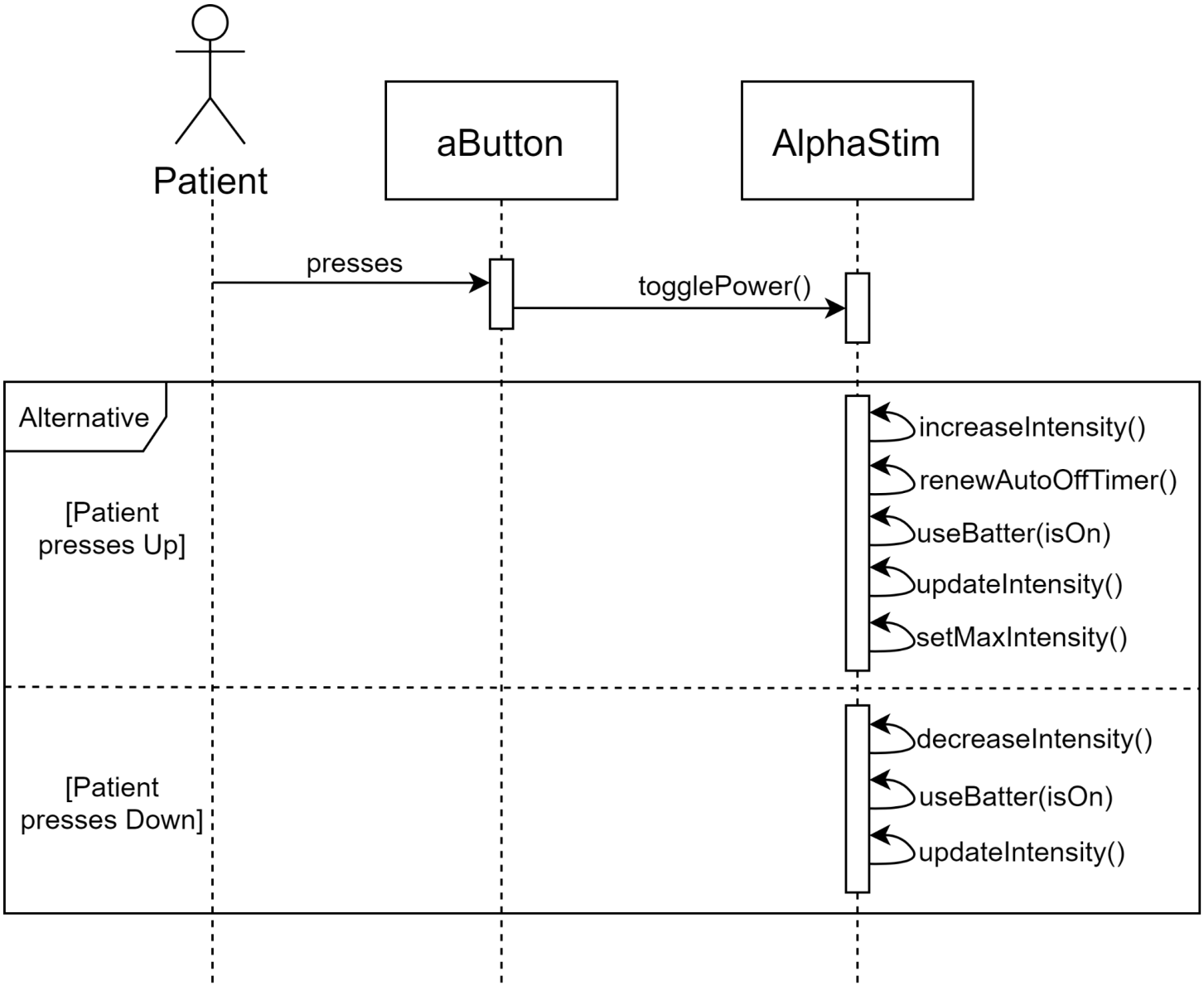
The patient begins by pushing the device's power button, which will invoke the togglePower() function. Thereafter, the patient will apply both electrodes to their skin. After attaching the electrodes, the patient will adjust the frequency, waveform, timer duration, and current. Finally, the user will push the start button, causing the AlphaStim to execute the startTherapy() method. Once the therapy session starts, the auto-off timer will be stopped.



UML Sequence Diagram for UC2

Starting from the top left of the UML Sequence Diagram:

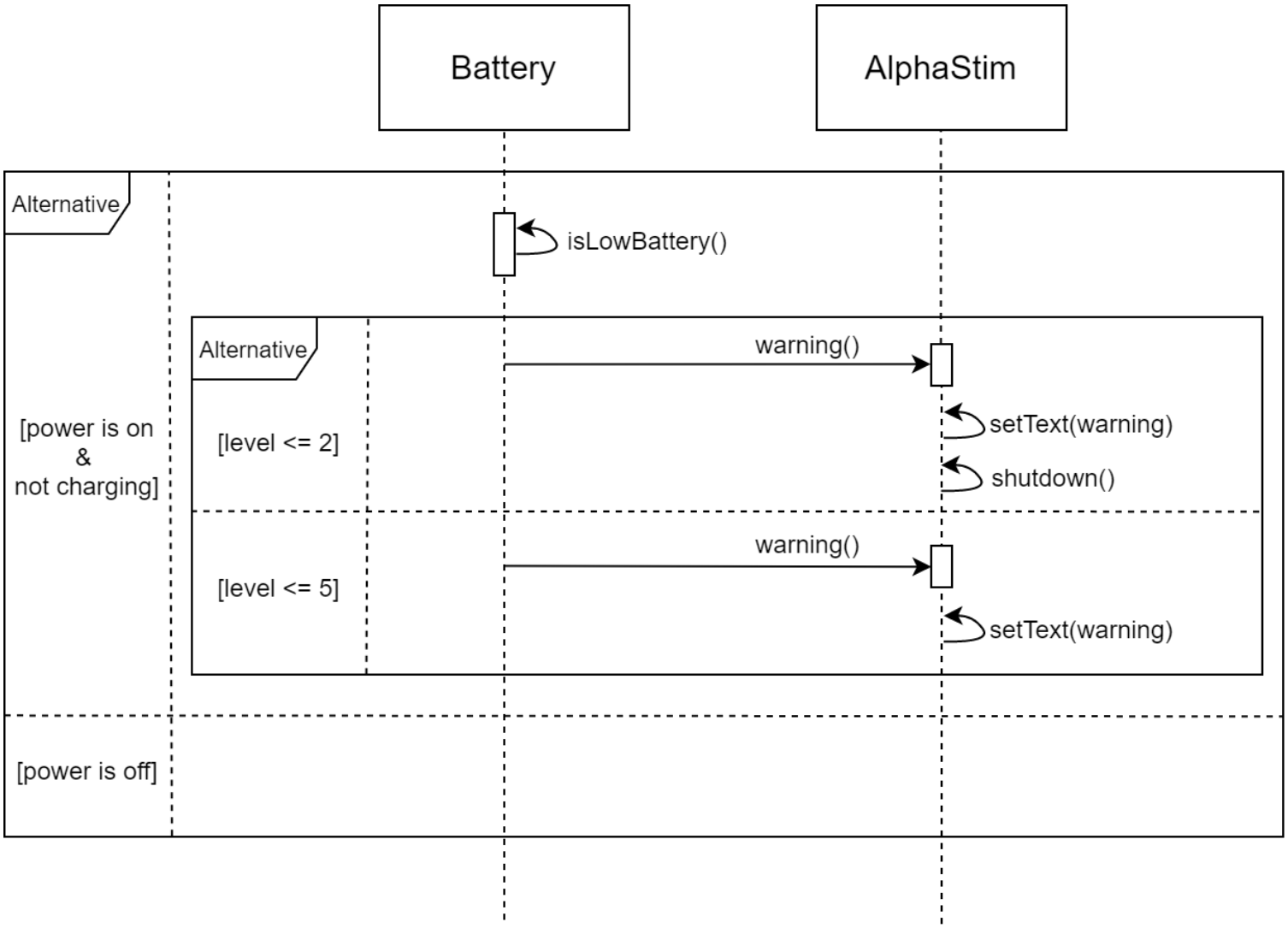
The patient begins by pushing the device's power button, which will invoke the turnOn() function. If the patient presses the up button, the intensity will increase, the auto-off timer will reset, the battery will be in use, the intensity will be updated on the UI, and a new max intensity will be set. If the patient presses the down button, the intensity will decrease, the battery will be in use, and the intensity will be updated on the UI.



UML Sequence Diagram for UC3

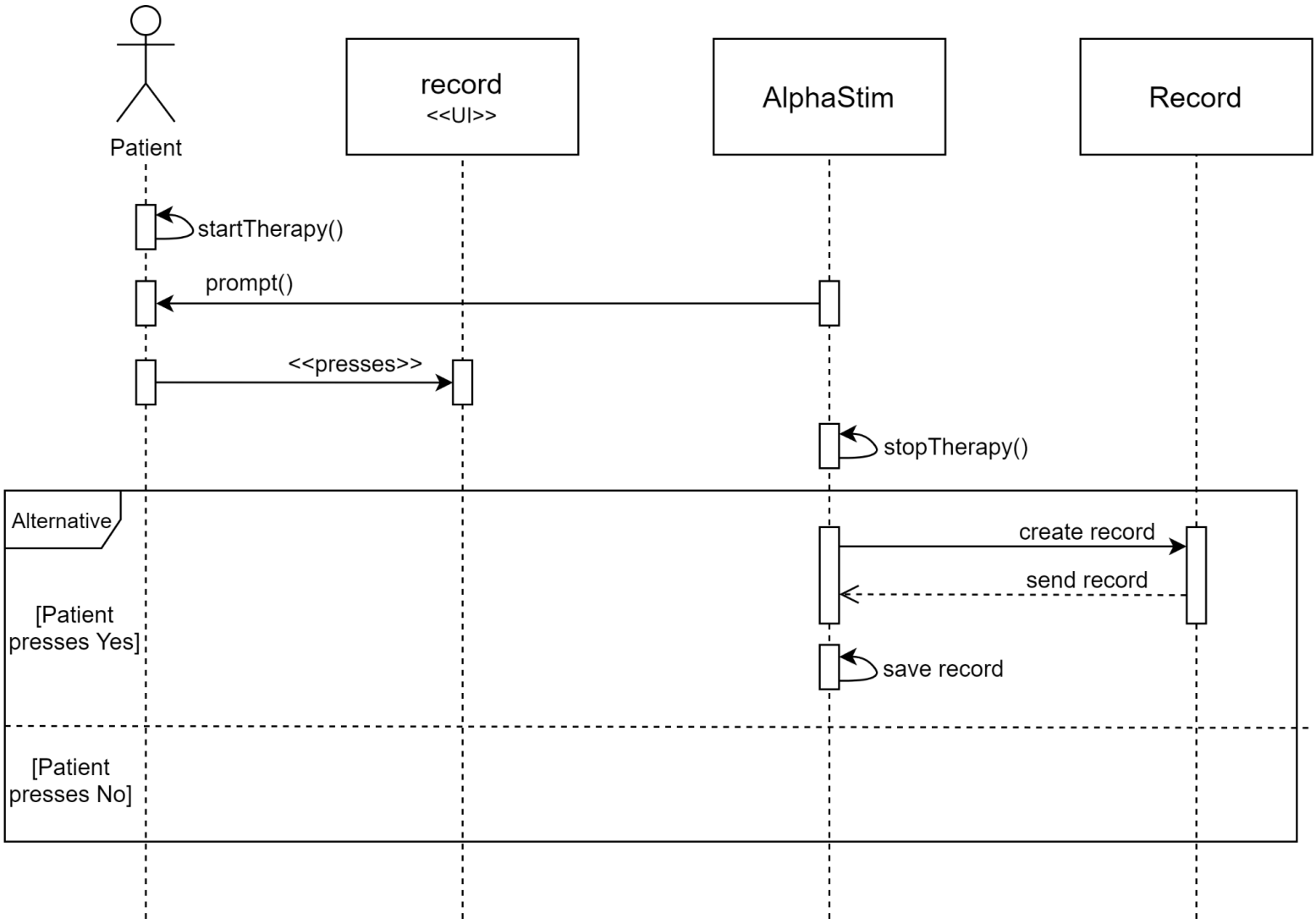
Starting from the top left of the UML Sequence Diagram:

If the power of the AlphaStim is on, the battery will check to see if its power level is low. If the battery level is less than or equal to 2%, the AlphaStim will be alerted, and the patient will get a low battery warning (via the UI). The AlphaStim will then be turned off. If the battery level is less than or equal to 5%, the AlphaStim will be alerted, and the patient will get a low battery warning (via the UI). If the power is off, nothing will happen.



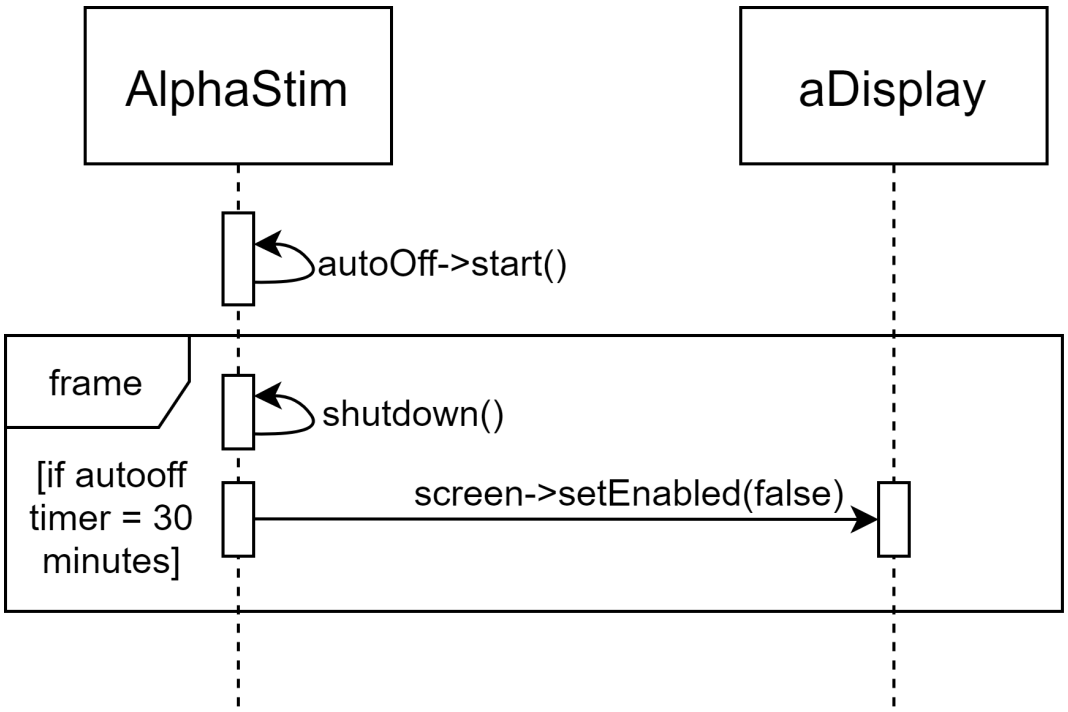
UML Sequence Diagram for UC4

The patient first begins by starting a therapy session. The patient can choose to record the session after a prompt from the AlphaStim. If the patient chooses to record a session, a record is made and saved after the therapy session is complete.



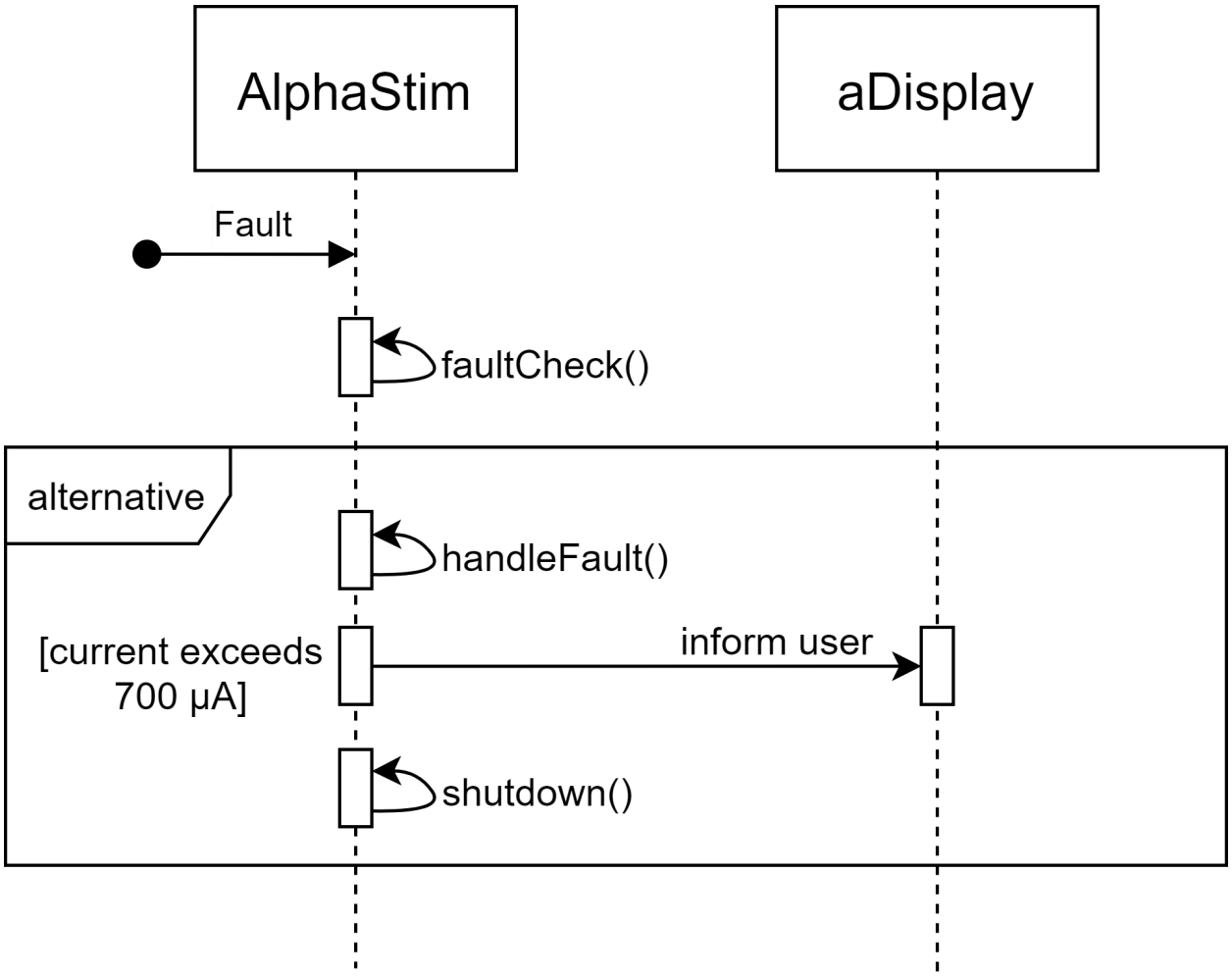
UML Sequence Diagram for UC5

The AlphaStim checks if it has been inactive for 30 minutes. If the AlphaStim has been inactive for 30 minutes, it will shut down and the UI will be changed to show that the device is turned off.



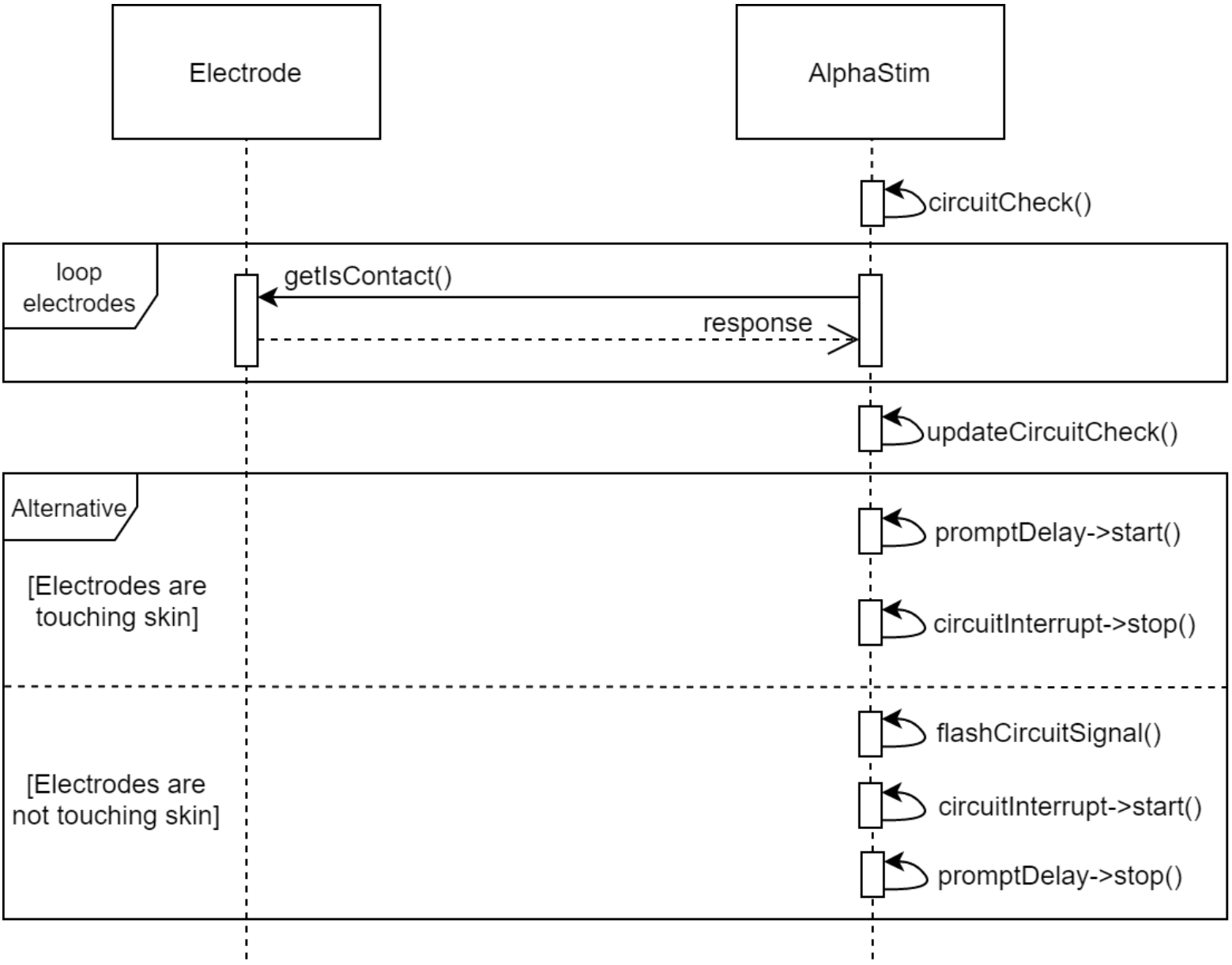
UML Sequence Diagram for UC6

If the AlphaStim receives a fault signal, it will determine the device's present current. If the current exceeds 700μa, the patient will be notified via the user interface (UI), and the AlphaStim will be permanently disabled.



UML Sequence Diagram for UC7

First, the AlphaStim device checks if both the electrodes are touching the patient's skin. If both electrodes are touching the patient's skin, a prompt delay timer will start to simulate a natural delay for when the device prompts the user to record a therapy session. Secondly, the warning symbol from the user interface (UI) is removed. If one or both of the electrodes are not touching the skin, a flashing warning symbol is displayed on the user interface (UI), a circuit interrupt warning is presented on the UI, and the prompt delay timer is stopped because there cannot start a therapy session if one or both of the electrodes are not touching the patient's skin.



TRACEABILITY MATRICES

Requirements to Use cases

Requirement ID	Requirement	Use Case
R001	On/Off switch	UC1: Starting the Therapy session
R002	Check when electrodes are in contact with the skin	UC7 : Check contact with skin
R003	Three Frequency Options	UC1: Starting the Therapy session
R004	Three Waveform Options	UC1: Starting the Therapy session
R005	20, 40, 60 countdown cycles	UC1: Starting the Therapy session
R006	Treatment starts when electrodes touch skin	UC1: Starting the Therapy session
R007	0-500μa current control	UC2: Adjusting the Current
R008	30 minute auto off when not in use	UC5 : Auto sleep when not in use
R009	Battery charge indicator	UC3 : Receiving low battery alerts
R0010	Recording	UC4 : Recording a therapy session
R0011	Automatic and permanent disabling if current exceeds 700μa.	UC6 : Auto disable when current exceeds 700μa.

Use Cases to Tests

Each test continuously verifies expected versus actual state and passes if they match at every stage. Otherwise, the test fails. Debug logs are printed to the console.

Use Case ID	Use Case	Test (test name and description)
UC1	Starting the Therapy session	*Test Scenario 4 - Start Therapy.* Turns on the device, connects electrodes, then configures and starts a therapy session
UC2	Adjusting the Current	*Test Scenario 3 - Changes current intensity.* Turns on the device, verifies min and max intensity limits, then sets the intensity to 350µa.
UC3	Receiving low battery alerts	*Test Scenario 8 - Force low battery state.* Forces the battery to 6% then waits for the 5% and 2% warnings.
UC4	Recording a therapy session	*Test Scenario 6 - Test saving a therapy session.* The device connects electrodes and runs a session. Then saves the record and checks if the length of the records list has increased.
UC5	Auto sleep when not in use	*Test Scenario 7 - Test auto sleep when not in use.* Turns on the AlphaStim and waits for 30 seconds for the AlphaStim to turn off. In real life the duration would be 30 minutes, but in order to save time we opted for seconds.

UC6	Auto disable when current exceeds 700μa	*Test Scenario 9 - Simulate 700μa intensity fault.* Turns AlphaStim on, creates a fault which causes an intensity of 700μa, then the device handles the fault.
UC7	Check contact with skin	*Test Scenario 2 - Continuous circuit check.* Turns AlphaStim on, connects and disconnects the electrodes repeatedly. Observe the test circuit symbol.