Design Decisions

Design Patterns

- Singleton
    - There's a single class modelling the data stored on the insulin pump such as the battery level, insulin on board etc.
    - This is to create a single entity which can be modified by any part of the pump (in this case other .cpp files)

- Observer
    - QT's default way of handling events and updating UI's
    - Signals and Slots are used to dynamically update the UI which eliminates the need for polling

- MVC
    - The data class is the model, the Qt GUI .ui is the view, and the various screen .cpps are the controller

- Facade
    - Logging class has abstract methods from its main logging method (info(), warn(), error() vs log()) to simplify logging different types of messages.

We chose to use these design patterns because they allow for modularity which allows for easier collaboration between group members since event handling can be done within each UI class and data is shared between a common "database" class that can be accessed by all users. Additionally, this modularity allows us to easily extend the feature set of the simulated insulin pump which was a common scenario during development.

Simulated Time

The insulin pump has a ratio of 1 real world second representing 5 simulated minutes. Values on the insulin pump will be updated every real world second. Logic also runs every one real world second. QTimers are used to run logic once per second.

Explanation of different state machine diagram states:

Program State Machine

- Insulin Pump Off
  - This represents the state of the program where the insulin pump is off. This would occur when either the user first starts the program or the user has pressed down and held the power button to turn off the insulin pump.
- Home Screen
  - This represents the state of the program where the user is on the home screen of the insulin pump simulation. They would reach this state by clicking either on the Home Screen button in the debug section or after entering the PIN during the regular boot process.
- Options Screen
  - This represents the state of the program where the user is on the home screen of the insulin pump simulation. They would reach this state by clicking either on the Options Screen button in the debug section or after clicking the Options button in the debug section.
- Bolus Screen
  - This represents the state of the program where the user is on the home screen of the insulin pump simulation. They would reach this state by clicking either on the Bolus Screen button in the debug section or after clicking the Options button in the debug section.
- Insulin Pump Asleep Screen
  - This represents the state of the program where the user quickly presses and releases the power button. This would put the insulin pump into sleep mode, where logic is still running.

Bolus State Machine

- Calculator
  - This represents the state where the insulin pump displays the bolus calculator. The user can enter this state by pressing on the Bolus button in the home screen or the bolus screen button in the debug section of the screen.
- Set Time
  - This represents the state where the insulin pump displays options for altering the delivery of the extended bolus. A user can get to this state by checking off the extended bolus checkbox in the 2nd part of the bolus calculator.
- Confirm start

    - This represents the state where the user confirms the provided inputs for the extended bolus and the extended bolus is being delivered. This is done by pressing the confirm button to deliver the extended bolus.
- Pumping
    - This represents the state of the insulin pump where the extended bolus or manual bolus is currently delivering insulin. This state occurs immediately after pressing the confirm start button.
- Bolus Paused
    - This represents the state of the insulin pump where the “Stop insulin delivery” button is pressed in the options menu to stop all insulin delivery.
- Stopped
    - This represents the state of the insulin pump where the given manual bolus or extended bolus has been completely delivered.