Постановка задачи

Необходимо разработать UML-диаграммы для программы управления демодулятором. Программа предназначена для наладчиков телекоммуникационного оборудования. Основные задачи программы: настройки частоты и контроля уровня сигналов. Связь устройства и компьютера осуществляется по технологии COM over USB. Для программы устройство будет доступно как COM-порт. Программа должна обладать графическим интерфейсом пользователя. Недопустимы ситуации, при которых интерфейс программы не отвечает на запросы пользователя. Разработка программы должна осуществляться на языке Java.

Основные параметры устройства:

* частота несущей - измеряется в МГц; представляется дробным числом; доступно на чтение и запись;
* уровень входного сигнала – измеряется в дБмкВ; представляется целым числом; измеряемый параметр (доступен только для чтения); измерение происходит один раз в секунду.

2. Диаграмма классов

Для удобства диаграмма классов всей системы разбита на две.

На рисунке представлена основная диаграмма классов. На диаграмме можно увидеть шаблон проектирования «модель-представление-контроллер» (англ. Model-View-Controller, MVC). В разрабатываемой системе он представлен классами DeviceView, DeviceModel и DeviceController. Класс DeviceView – представление, отвечающее за отображение данных устройства пользователю. Представление берет данные из модели – DeviceModel. При необходимости обработки пользовательских действий представление обращается к контроллеру – DeviceController, который меняет модель и осуществляет дополнительные действия. Так, например, в разрабатываемой системе контроллер отправляет новые настройки частоты на устройство с помощью класса AsyncCommandExecutor. Представление узнает об изменении модели с помощью шаблона «наблюдатель» (англ. observer).

На рисунке представлена диаграмма классов, описывающая классы ответственные за общение с устройством. Логика общения устройства инкапсулирована с помощью шаблона проектирования «команда» (англ. command) в наследниках абстрактного класса BaseCommand: GetFrequencyCommand (команда запроса частоты), SetFrequencyCommand (команда установки нового значения частоты) и GetSignalLevelCommand (команда получения текущего уровня сигнала). Контроллер DeviceController с помощью перечисленных выше команд осуществляет общение с устройством по средствам активного класса AsyncCommandExecutor, который исполняет команды в порядке поступления. Класс AsyncCommandExecutor – активный, т.е. исполняет свою логику в отдельном потоке. Это связано с тем фактом, что операции ввода и вывода длительные и могут привести к зависанию пользовательского интерфейса.

Диаграммы компонентов и размещения

На рисунке представлена диаграмма размещения компонентов системы. С помощью диаграммы видно, что компьютер и устройство соединены по технологии «COM over USB». На компьютере пользователя размещается компонент «Программа управления», который и описан в данной работе, а на устройстве располагается компонент «Прошивка» осуществляющий управление устройством (данный компонент в работе не рассматривается).

На рисунке представлена диаграмма компонентов программы «Программа управления». Система разбита на четыре компонента:

* GUI – компонент графического пользовательского интерфейса;
* DomainModel – доменная модель программы;
* CommunicationModule – компонент ответственный за прием и передачу сообщений устройству;
* JavaCore – библиотека языка Java.

Компонент GUI зависит от компонента DomainModel. Компонент DomainModel зависит от компонента CommunicationModule. Компоненты GUI, DomainModel и CommunicationModule зависят от компонента JavaCore.

Диаграмма вариантов использования

На рисунке представлена диаграмма вариантов использования программы. С системой взаимодействуют два актора: «Наладчик» и «Прошивка».

Наладчик может выполнять две задачи: настройку частоты и осуществлять контроль уровня сигнала.

Прошивка демодулятора сообщает о текущем состоянии устройства. Эти данные отображаются пользователю во время контрольного наблюдения за устройством.

5. Диаграмма взаимодействия

На рисунке представлена диаграмма взаимодействия системы, описывающая процесс изменения частоты пользователем. Пользователь меняет частоту, используя форму настроек на экране (объект класса DeviceView). Представление делегирует обязанности изменения частоты контроллеру DeviceController. Контроллер меняет данные в модели DeviceModel на новые. Модель оповещает своих «наблюдателей» об изменении, в числе которых находится и представление. После изменения модели, контроллер отправляет команду устройству на изменение частоты с помощью объекта класса AsyncCommandExecutor.

6. Диаграмма состояний

На рисунке представлена диаграмма состояний объектов класса DeviceController. Первое состояние – «Инициализация», попадая в которое объект запрашивает установленную частоту у устройства. Если данные о частоте были успешно получены, то дальше объект переходит в состояние «Обновление», при попадании в которое у устройства запрашиваются данные о текущем уровне сигнала. Если данные были успешно получены, то контроллер переходит в состояние «Простой», из которого он может выйти двумя путями – дождаться устаревания данных или изменения пользователем частоты.

7. Диаграмма деятельности

На рисунке представлена диаграмма деятельности, описывающая выполнение команды. В объектах класса AsyncCommandExecutor выстраивается очередь из команд, посылаемых устройству, для выполнения берется первая в очереди команда. Для отправки запроса на устройство используется объект класса SerialPort, который контролирует COM-порт. После отправки на устройства запроса начинается ожидание ответного сигнала. Если запрос выполнен успешно, посылается сигнал об успешном выполнении команды, иначе посылается сигнал об ошибке. Также если отправка и ожидание ответа длятся более 2 секунд, то запрос считается выполненным с ошибкой и посылается сигнал об ошибке во время выполнения запроса. Далее команда считается выполненной, и она удаляется из очереди. После этого поток выполнения команды считается оконченным.

Заключение