При проектировании ПО модуля УРОВ во внимание, основываясь на выбранной архитектуре системы, были приняты следующие факторы:

1. Наличие основного ядра системы, выполняющего рутинные работы по обработке компонентов системы – обновление модулей, обеспечение взаимодействия модулей друг с другом;
2. Наличие централизованного хранилища общей информации, которой оперируют модули – настройки УРОВ, общий доступ к хранилищам информации (память EEPROM, SD-карта);
3. Программное разделение представления данных от способа их отображения – будь то на встроенном экране, или на Конфигураторе для персональных компьютеров.

Вследствие вышесказанного, при проектировании ПО были выделены следующие компоненты, требующие реализации:

1. Класс настроек системы УРОВ (класс Settings) – обеспечивает доступ и манипулирование общей информацией в рамках системы, такой как уставки количества импульсов, данные о количестве срабатываний штанги и пр.;
2. Класс обмена информацией с внешним миром посредством USB (класс CommandHandler) – предоставляет собой обезличенный текстовый интерфейс общения с модулем УРОВ из внешнего мира, посредством текстовых команд определённого формата. Данный подход позволяет, во-первых, в любой момент значительно расширить функционал, без переделывания архитектуры проекта, и, во-вторых – обеспечивает удобный и понятный интерфейс взаимодействия по любым каналам связи, т.к. является прикладным интерфейсом общения, независящим от способа передачи информации;
3. Классы работы с хранилищами данных – EEPROM-память и SD-карта, предоставляющие набор программных методов для манипулирования информацией, которую необходимо сохранять на постоянной основе – уставки, лог-файлы, любая другая информация, требующая энергонезависимого хранения;
4. Классы представления информации на встроенном TFT-экране, выполненные в виде программных интерфейсов, основанных на абстрактном классе виртуального экрана. Путём добавления конкретной реализации конкретного экрана в класс-диспетчер – получаем гибко расширяемый функционал по оснащению модуля УРОВ графическим интерфейсом пользователя;
5. Класс обработки прерываний от энкодера штанги, который обеспечивает реализацию фиксирования списка прерываний как во внутреннюю, так и в постоянную память контроллера, и, при необходимости, посредством программных интерфейсов – уведомление программных модулей о произошедшем событии срабатывания штанги;
6. Класс фиксации параметров тока, обеспечивающий сбор информации с аналогово-цифрового преобразователя в течение заданного промежутка времени, система хранения данных которого реализована в виде кольцевого буфера;
7. Класс работы с аналого-цифровым преобразователем, который обеспечивает непрерывный сбор информации с нескольких каналов АЦП с сохранением этих данных в несколько программных буферов внутренней памяти микроконтроллера, на основе которых (данных) можно предпринимать дальнейшие действия по обработке и анализу информации;
8. Вспомогательные классы для представления информации в виде графиков на встроенном экране (пространство имён Drawing) – позволяющие по исходному набору данных сформировать тот или иной график, с выводом его на экран;
9. Класс работы с интерфейсом RS-485, реализующий программный протокол обмена информацией с внешним модулем системы;
10. Вспомогательный класс для сравнения эталона с графиком срабатывания – выделение этой сущности в отдельный модуль позволяет легко и независимо от других компонентов системы менять алгоритмы сравнения;
11. Вспомогательные классы для работы с электронными компонентами системы, такие как часы реального времени и пр.;
12. Класс обратной связи (Feedback), предназначенный, путём вызова его публичных методов, для информирования пользователя о произошедших в системе изменениях, путём сигнализирования светодиодами модуля;
13. Программный код работы с ватчдогом – концепцией, при зависании микроконтроллера перезапускающей его питание;
14. Ядро системы – набор функций, соединяющих работу всех компонентов системы воедино.

По результатам проектирования ПО, на выходе был получен набор концепций, необходимых к реализации, каждая в своём пространстве (файлах исходного кода).

При реализации концепций было принято во внимание, что сходные по внутреннему функционалу части концепций должны иметь одинаково называющиеся публичные методы. Так, для пуска модуля системы в работу – используется мнемоника названия метода begin или setup, для обновления внутреннего состояния модуля – название update, для отрисовки состояния на экране – название draw и т.п. Все интерфейсы обратного вызова (callback) – в обязательном порядке описываются через структуру, все имена методов которой имеют в префиксе предлог «On», означающий принадлежность к обработке какого-либо события, например – OnInterruptRaised – обработчик события «фиксирование прерывания по срабатыванию штанги».