При разработке программной архитектуры во внимание принималось:

1. Список требований заказчика;
2. Анализ требований, проведённый на основании пункта 1;
3. Принятые в IT подходы по реализации программной архитектуры, применимые к проекту.

Основой архитектуры системы была принята модульная ООП-система, где каждый программный модуль представляет собой класс (или набор классов), предназначенный для реализации части алгоритма работы всей системы по принципу «чёрный ящик», предоставляя наружу интерфейсы для общения с модулем.

Для обмена информацией и взаимодействия программных модулей между собой, наряду с устоявшимися подходами в виде открытых публичных методов класса – было принято решение использовать событийный интерфейс на основе функций обратного вызова (callback) и программных интерфейсов, что, по определению, позволяет расширить универсальность взаимодействия программных модулей друг с другом.

Вследствие одноядерности исполнения используемых электронных компонентов (микроконтроллер), а также отсутствия истинной многозадачности в используемых электронных компонентах – одной из парадигм проектируемой программной архитектуры проекта была предложена концепция неблокирующей обработки, когда каждому модулю поочерёдно предоставляется программное время на обновление своего внутреннего состояния, и к каждому программному модулю предъявляется требование о недопустимости долгих блокирующих операций, вследствие чего вытекает архитектурное требование о реализации программных модулей, требующих сложной алгоритмической работы, в виде finite state machine (конечных автоматов с заранее известным набором состояний). Подобный подход избавляет архитектуру от недостатков отсутствия многозадачности, однако – накладывает определённые требования как к реализации, так и к квалификации исполнителей, поскольку асинхронный подход – в реализации не самый простой.

Как итог, архитектура проекта представляет собой:

1. Набор модулей;
2. Центральное ядро, занятое обработкой и модулей, обеспечивающее своевременное предоставление модулям квантов времени, необходимых для поддержания их внутреннего состояния;
3. Набор интерфейсов и методов, связывающих модули между собой.